COMPUTER METHODS FOR AEROSPACE TRAJECTORY OPTIMIZATION

Written by Columbus David Eagle Jr.

August 2020

Table of Contents

Introduction ……..…..3

CMATO 1 – Single Maneuver, Finite-Burn Trajectory Optimization ……..4

CMATO 2 – Two Maneuver, Finite-Burn Trajectory Optimization ……...28

CMATO 3 – Low-Thrust LEO-to-GEO Trajectory Optimization ……..……53

CAMTO 4 – Finite-Burn, Earth Orbit Rendezvous Trajectory Optimization …….……69

CMATO 5 – Atmospheric Reentry Trajectory Optimization …….……89

CMATO 6 – Aero-assist Trajectory Optimization ……...…….108

CMATO 7 – Finite-Burn De-orbit Trajectory Optimization …….……129

CMATO 8 – Lunar Ascent Trajectory Optimization …….……151

CMATO 9 – Impulsive Trans-Lunar Trajectory Optimization …….……172

CMATO 10 – Finite-burn Trans-Lunar Trajectory Optimization …….188

CMATO 11 – Ballistic Interplanetary Trajectory Optimization ……211

CMATO 12 –Earth-to-Mars End-to-End Mission Design …….……230

CMATO 13 – Interplanetary Gravity-Assist Trajectory Optimization ……..265

CMATO 14 – Low-Thrust Interplanetary Trajectory Optimization ……..291

CMATO 15 – Interplanetary Trajectory Correction Maneuver Optimization …….……..321

CMATO 16 – Optimal Finite-Burn Earth Orbit-to-Interplanetary Injection …….……346

CMATO 17 – Low-thrust Earth Escape Trajectory Optimization …….……376

Appendices

A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System ……...391

B – Trajectory Modeling in the Flight Path System ……..…..400

C – Cartesian Equations of Motion …….407

D – B-Plane Geometry, Coordinates and Targeting …….……..413
..
E – Impulsive De-orbit from Earth Orbit ……418

F – Computing an Asteroid or Comet Ephemeris …….……..423

G – Aerospace Trajectory Coordinates and Time Systems ……428

H – Impulsive Interplanetary Injection from a Circular Earth Orbit …….…….440

I – Numerical Solutions of Lambert's Problem …….…….447

J – Sparse Optimization Suite Configuration File ……..……..……..452

K – Example Fortran Subroutines ……..……454

Introduction

This document describes the implementation of Fortran computer programs for solving practical problems in aerospace trajectory optimization. All programs are written in Fortran and utilize the *Sparse Optimization Suite (SOS)* software distributed by Applied Mathematical Analysis (*AMA*).

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

The *CMATO* software consists of Fortran routines that perform the following tasks.

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

SOS will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method.

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010, and these documents.

John T. Betts and Paul D. Frank. A Sparse Nonlinear Optimization Algorithm. *Journal of Optimization Theory and Applications*, 82(3):519–541, September 1994.

John T. Betts and William P. Huffman. Application of Sparse Nonlinear Programming to Trajectory Optimization. *AIAA Journal of Guidance, Control, and Dynamics*, 15(1):198–206, January–February 1992.

John T. Betts and William P. Huffman. Path Constrained Trajectory Optimization Using Sparse Sequential Quadratic Programming, *AIAA Journal of Guidance, Control, and Dynamics*, 16(1):59–68, January–February 1993.

I sincerely thank Dr. John Betts for sharing his experience and expertise in engineering optimization.

CMATO 1 – Single Maneuver, Finite-Burn Trajectory Optimization

This *CMATO* application is a Fortran computer program named `oneburn_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve an Earth orbit transfer trajectory optimization problem. The software models the trajectory as a single, finite-burn propulsive maneuver followed by a user-defined, time-bounded final coast phase. This computer program attempts to maximize the final spacecraft mass. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass.

The important features of this scientific simulation are as follows:

- single, continuous thrust orbital maneuver
- variable inertial attitude steering
- constant propulsive thrust magnitude
- modified equinoctial equations of motion with oblate Earth gravity model
- user-specified final coast phase

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `oneburn_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method.

The `oneburn_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `oneburn_sos` software is "data-driven" by a text file created by the user. This text file should be simple ASCII format with no special characters.

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly.

The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in bold courier font and all explanations are in times font. This example attempts to optimize the maneuver required to transfer a spacecraft from a circular low Earth orbit (LEO) to a typical elliptical geosynchronous transfer orbit (GTO).

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*********************************************
** earth-orbit trajectory optimization
** single finite-burn maneuver with final coast
** program oneburn_sos
** leo2gto.in - February 20, 2012
*********************************************
```

The first three numerical inputs define the initial mass prior to the propulsive maneuver, and the thrust magnitude and specific impulse of the upper stage or spacecraft propulsion system.

```
initial spacecraft mass (kilograms)
10000.0
thrust magnitude (newtons)
99200.0

specific impulse (seconds)
450.0
```

This next integer input defines the type of initial guess for the propulsive maneuver.

```
******************************
type of propulsive initial guess
******************************
1 = thrust duration
2 = delta-v
-------------------
2
```

The next two numeric inputs define either the user's initial guess for the delta-v magnitude or the maneuver duration and should be consistent with the previous input.

```
initial guess for delta-v (meters/second)
2800.0

initial guess for thrust duration (seconds)
550.0
```

The next two inputs define the lower and upper bounds for the thrust duration. These inputs are required for either type of propulsive initial guess.

```
lower bound for thrust duration (seconds)
0.01

upper bound for thrust duration (seconds)
10000.0
```

The next section of the input data file defines the characteristics of a final coast phase that follows the propulsive maneuver. These three inputs define an initial guess for the coast duration as well as lower and upper bounds on the coast duration.

```
**************
coast maneuver
**************

initial guess for coast duration (seconds)
20.0

lower bound for coast duration (seconds)
1.0

upper bound for coast duration (seconds)
2000.0
```

The next six inputs define the classical orbital elements of the initial park orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
6563.14d0

orbital eccentricity (non-dimensional)
0.0
orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
0.0d0

true anomaly (degrees)
0.0
```

This next integer input allows the user to define the type of initial orbit constraints to use during the simulation.

```
*******************************
initial orbit constraint options
*******************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-------------------------------------------
3
```

The next six inputs define the classical orbital elements of the final mission orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
***************
* FINAL ORBIT *
***************

semimajor axis (kilometers)
24364.8d0

orbital eccentricity (non-dimensional)
0.73062206d0

orbital inclination (degrees)
26.3355d0

argument of perigee (degrees)
270.0d0

right ascension of the ascending node (degrees)
0.0d0

true anomaly (degrees)
0.0d0
```

This next integer input allows the user to define the type of final orbit constraints to use during the simulation.

```
*****************************
final orbit constraint options
*****************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all final orbital elements
3 = option 2 with unconstrained true longitude
-------------------------------------------
1
```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will use a J_2 gravity model in the spacecraft equations of motion.

```
************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate gravity model
------------------------
2
```

This next input defines the type of initial guess to use. Please see the technical discussion section for information about how the first option is modeled. Option 2 requires either a binary restart file created from a previous run using either initial guess option 1 or an updated binary restart file. This feature is described in the next two sections.

```
************************
* initial guess options *
************************
 1 = numerical integration
 2 = binary data file
---------------------
1
```

If the user elects to use a binary data file (option 2 above) for the initial guess, the following text input specifies the name of the file to use.

```
name of binary initial guess data file
leo2gto.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess option 1, the software will create a binary restart file. For initial guess option 2, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*****************************
* binary restart file option *
*****************************

create/update binary data file (yes or no)
no
```

This next input specifies the type of solution data file to create.

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
--------------------------
2
```

For options 2 or 3, this input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
100
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
leo2gto.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

This input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

This input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the *Sparse Optimization Suite* NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the *Sparse Optimization Suite* optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of numerical information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the *Sparse Optimization Suite* differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of numerical information.

```
***************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string **no**.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of inputs allow the reading and writing of configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `oneburn_sos` software. Please consult *Appendix J – Typical Sparse Optimization Suite Configuration File*, for more information about this type of file.

```
************************************
* optimal control configuration options
************************************

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
leo2gto_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
leo2gto_config1.txt
```

Optimal control solution

The following is the optimal control solution for this example. This simulation used variable attitude steering during the propulsive maneuver, and a final bounded coast. The output includes the time and orbital characteristics at the beginning and end of the propulsive maneuver. This example optimizes the maneuver required to transfer from a circular low Earth orbit (LEO) to a typical elliptical geosynchronous transfer orbit (GTO).

```
program oneburn_sos
===================

input file ==> leo2gto.in

oblate earth gravity model

------------------------
beginning of finite burn
------------------------

mission elapsed time     00:00:00.000

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.656314000000D+04    0.303099251680D-16    0.285000000000D+02    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.343626272476D+03    0.284333562833D+03    0.284333562833D+03    0.881915957810D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.164199672837D+02    -.581964995764D+04    -.303417392626D+04    0.656314000000D+04
```

```
     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 0.772230207356D+01    -.501755544829D+00    0.920593794451D+00    0.779315089527D+01

 ------------------
 end of finite burn
 ------------------

 mission elapsed time     00:03:29.965

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.243647798517D+05    0.730621883714D+00    0.263354841203D+02    0.269999803084D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.125070318737D-03    0.159279650912D+02    0.285927768175D+03    0.630817068563D+03

     rx (km)               ry (km)               rz (km)               rmag (km)
 0.183083308977D+04    -.574950274971D+04    -.284601509915D+04    0.667147162297D+04

     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 0.100245469314D+02    0.145694376803D+01    0.721178679773D+00    0.101555071272D+02
```

The following program output is the final spacecraft mass, the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v.

```
final mass             5280.17375040397      kilograms

propellant mass        4719.82624959603      kilograms

thrust duration        209.965300814227      seconds

delta-v                2818.25253032923      meters/second
```

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node or user-defined step size.

This section of the numerical results summarizes the time and orbital conditions at the beginning and end of the final coast.

```
 ---------------------------
 beginning of coast maneuver
 ---------------------------
 mission elapsed time     00:03:29.965

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.243647798517D+05    0.730621883714D+00    0.263354841203D+02    0.269999803084D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.125070318734D-03    0.159279650912D+02    0.285927768175D+03    0.630817068563D+03

     rx (km)               ry (km)               rz (km)               rmag (km)
 0.183083308977D+04    -.574950274971D+04    -.284601509915D+04    0.667147162297D+04

     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 0.100245469314D+02    0.145694376803D+01    0.721178679773D+00    0.101555071272D+02

 ---------------------
 end of coast maneuver
 ---------------------

 mission elapsed time     00:03:30.965

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.243648000000D+05    0.730622060000D+00    0.263355000000D+02    0.270000000000D+03
```

```
   raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
0.360000000000D+03    0.160144835450D+02    0.286014483545D+03    0.630817851038D+03

     rx (km)                ry (km)               rz (km)               rmag (km)
0.184085640569D+04    -.574804194749D+04    -.284529200486D+04    0.667266252177D+04

     vx (kps)               vy (kps)              vz (kps)              vmag (kps)
0.100220828732D+02    0.146465973235D+01    0.725009431797D+00    0.101544577367D+02

coast duration         1.00000000000151      seconds
                      1.666666666669177E-002 minutes
```

Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* can be verified by numerically integrating the orbital equations of motion with the *SOS*-computed initial park orbit conditions and the optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the optimal unit thrust vector solution from the *SOS* solution. This part of the `oneburn_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion using the optimal controls.

The following is a display of the final solution computed using this *explicit* numerical integration method.

```
========================================
verification of optimal control solution
========================================
------------------
end of finite burn
------------------

mission elapsed time     00:03:29.965

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.243647798517D+05    0.730621883714D+00    0.263354841203D+02    0.269999803084D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.125070318737D-03    0.159279650912D+02    0.285927768175D+03    0.630817068563D+03

     rx (km)                ry (km)               rz (km)               rmag (km)
 0.183083308977D+04    -.574950274971D+04    -.284601509915D+04    0.667147162297D+04

     vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 0.100245469314D+02    0.145694376803D+01    0.721178679773D+00    0.101555071272D+02

final mass              5280.17375040377      kilograms

propellant mass         4719.82624959623      kilograms

thrust duration         209.965300814227      seconds

delta-v                 2818.25213904842      meters/second

final mission orbit
-------------------

mission elapsed time     00:03:30.965

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.243648000037D+05    0.730622059923D+00    0.263355000035D+02    0.270000000226D+03
```

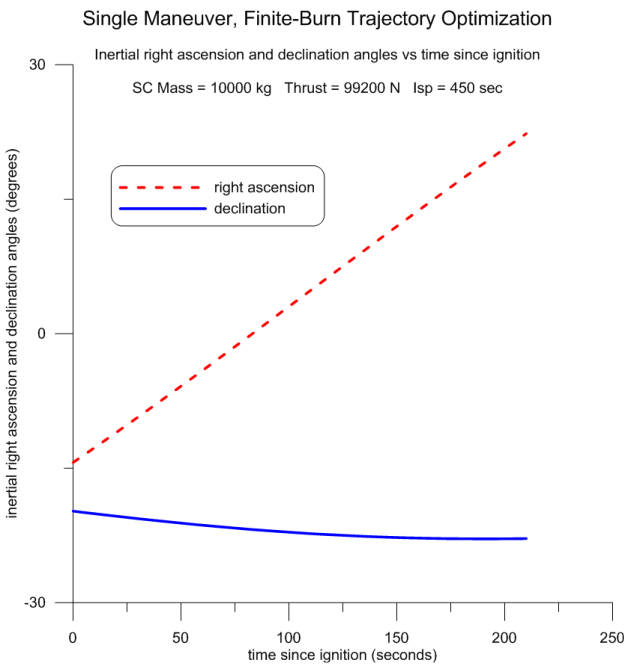
raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.359999999997D+03	0.160144833255D+02	0.286014483551D+03	0.630817851180D+03

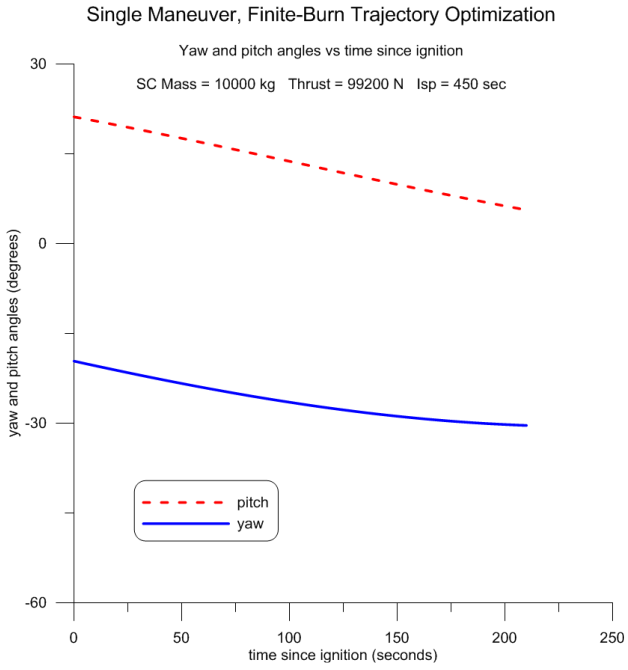
rx (km)	ry (km)	rz (km)	rmag (km)
0.184085640604D+04	-.574804194714D+04	-.284529200507D+04	0.667266252165D+04

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.100220828707D+02	0.146465974729D+01	0.725009439586D+00	0.101544577369D+02

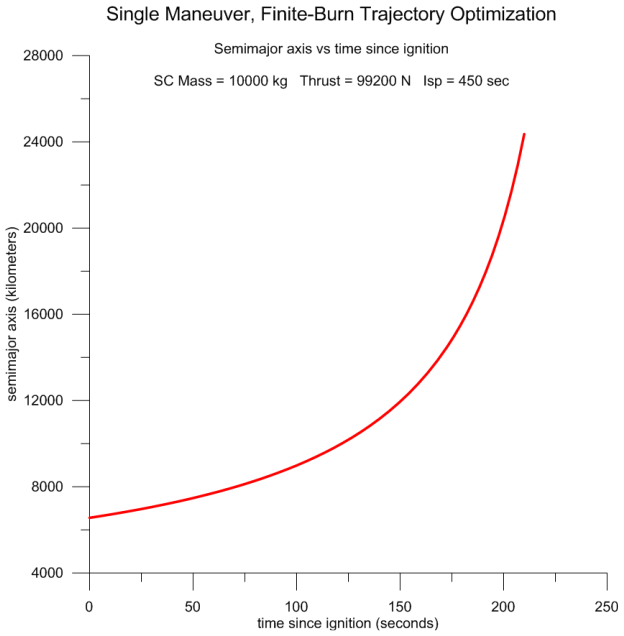
In additional to the user-defined solution output file, the `oneburn_sos` program will create two comma-separated-variable data files named `orbits.csv` and `maneuver.csv`. The first file contains position vectors of the initial and final orbits normalized with respect to the radius of the Earth. The second data file contains trajectory conditions starting at ignition and ending at burnout of the propulsive maneuver.

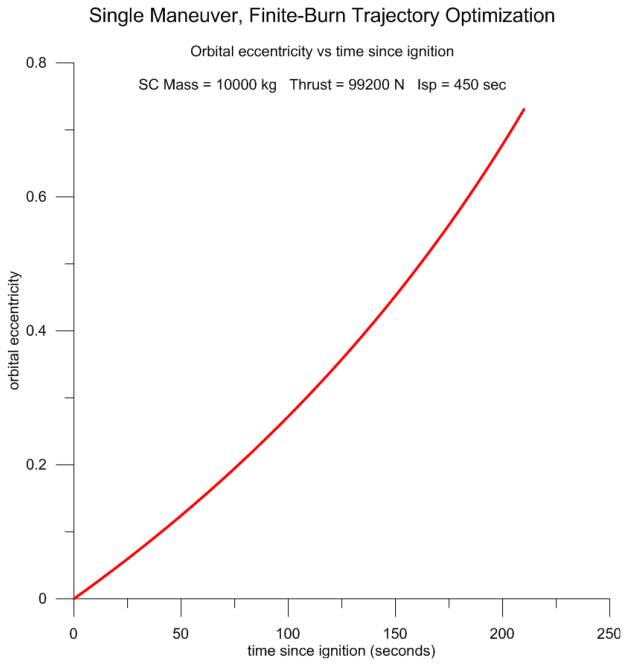
The following plots illustrate the evolution of the inertial right ascension, declination and pitch and yaw angles during this finite-burn maneuver.



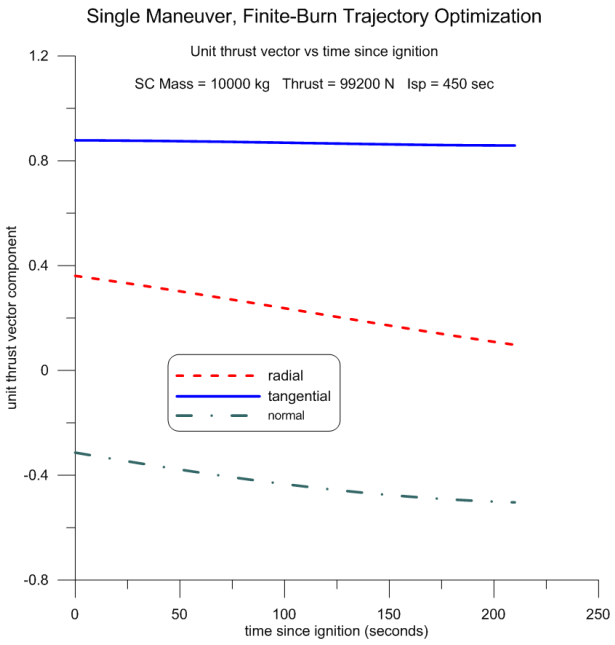


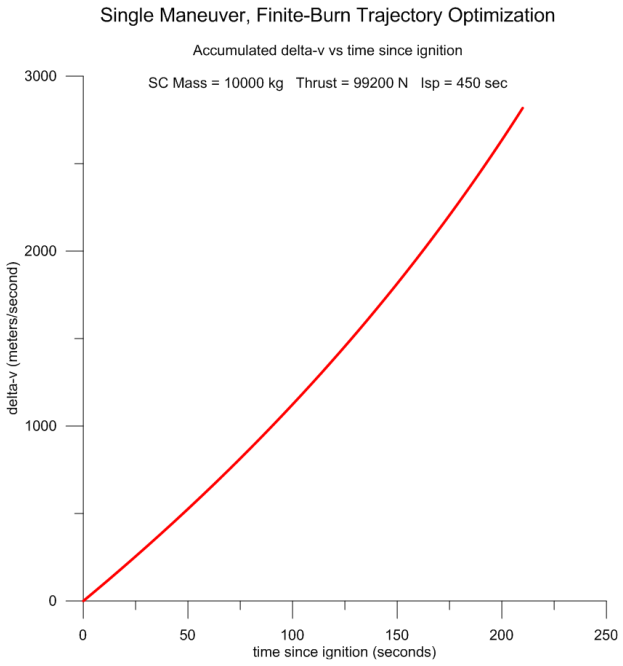
The next two plots illustrate the evolution of the semimajor axis and orbital eccentricity of the transfer orbit during this finite-burn maneuver.



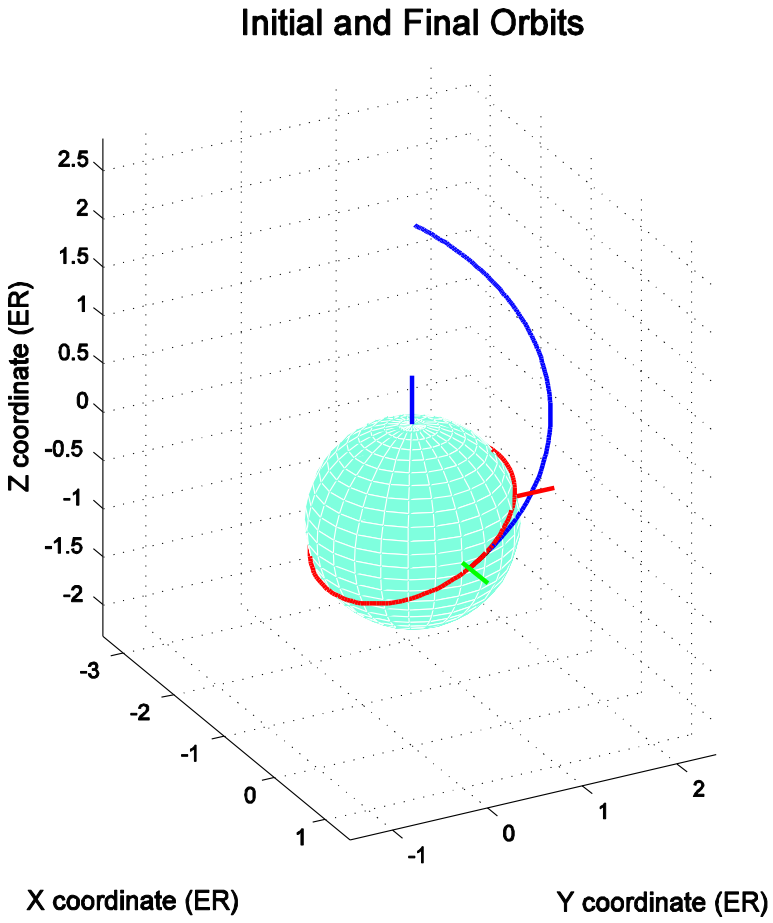


The final two plots illustrate the behavior of the radial, tangential and radial components of the unit thrust vector, and the accumulated delta-v during the propulsive maneuver.





The following is a 3-D graphics display of the initial (red trace) and final orbits (blue trace). The coordinates are shown in the units of Earth radii (ER).



Creating an initial guess

The software allows the user to input either a delta-v or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. For either type of initial guess, the user should also provide lower and upper bounds for the total thrust duration.

An estimate of the thrust duration can be determined from the following expression

$$t_d = \frac{I_{sp} m_p g}{F} = \frac{m_p V_{ex}}{F}$$

The propellant mass required for a given ΔV is a function of the initial (or final) mass of the spacecraft and the exhaust velocity as follows

$$m_p = m_i \left(1 - e^{\frac{-\Delta V}{V_{ex}}} \right) = m_f \left(e^{\frac{\Delta V}{V_{ex}}} - 1 \right)$$

In these equations

$m_i =$ initial mass $\quad m_f =$ final mass $\quad m_p =$ propellant mass

$V_{ex} =$ exhaust velocity $= g I_{sp} \quad I_{sp} =$ specific impulse

$\Delta V =$ impulsive velocity increment $\quad F =$ thrust $\quad g =$ acceleration of gravity

The software requires an initial guess for the thrust duration. The user should also provide lower and upper bounds for the total thrust duration. All inputs should be in seconds. If the *Sparse Optimization Suite* cannot find a feasible solution, try increasing the guess for thrust duration.

The software uses tangential steering to generate an initial guess for the trajectory. For tangential thrusting, the unit thrust vector in the modified equinoctial frame is simply $\mathbf{u}_T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$. Please note this type of steering creates a *coplanar* initial guess. It works best when the initial and final orbits are nearly coplanar.

The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine for this aerospace trajectory optimization problem. These program options create an initial guess from the numerical integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 2` program option tells the program to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

Binary restart data files can also be used to initialize a `oneburn_sos` simulation. A typical scenario is

1) create a binary restart file from a converged and optimized simulation
2) modify the original input file with slightly different spacecraft characteristics, propulsive parameters or perhaps final mission targets and/or constraints
3) use the previously created binary restart file as the initial guess for the new simulation

This technique works provided the two simulations are not dramatically different. Sometimes it may be necessary to make successive small changes in the mission definition and run multiples simulations to eventually reach the final desired solution.

Problem setup

This section of the document provides details about the `oneburn_sos` software implementation. It defines such things as point and path constraints (boundary conditions), bounds on the dynamic variables, and the performance index or objective function.

Point functions – initial orbit constraints

The software allows the user to select one of the following initial orbit constraint options

1) constrain semimajor axis, eccentricity and inclination
2) constrain all initial orbital elements
3) option 2 with unconstrained true longitude

For option 1, the initial orbit inclination is constrained by enforcing

$$\sqrt{h^2 + k^2} = \tan\left(\frac{i}{2}\right)$$

where *i* is the initial orbit inclination.

If the initial orbit is circular, the computer program enforces the two equality constraints $f = 0$ and $g = 0$.

Otherwise, for an elliptical initial orbit, the single equality constraint $\sqrt{f^2 + g^2} = e$ is enforced, where *e* is the initial orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the initial modified equinoctial orbital elements as follows

$$p_L = p_U = p_i \qquad f_L = f_U = f_i \qquad g_L = g_U = g_i$$

$$h_L = h_U = h_i \qquad k_L = k_U = k_i$$

Option 3 is identical to option 2 with the initial true longitude unbounded.

In optimal control terminology, these derived constraints or boundary conditions are called *point functions*.

Performance index – maximize final spacecraft mass

The objective function or performance index *J* for this simulation is the mass of the spacecraft at burnout or termination of the propulsive maneuver. This is simply $J = m_f$.

The value of the `maxmin` indicator in the *Sparse Optimization Suite* algorithm tells the software whether the user wants to minimize or maximize the performance index. The spacecraft mass at the initial time is fixed to the user-defined initial value.

Path constraint – unit thrust vector scalar magnitude

For a *variable steering* trajectory, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as follows

$$\left|\mathbf{u}_T\right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

Point functions – final mission orbit constraints

The software allows the user to select one of the following final orbit constraint options

1) constrain semimajor axis, eccentricity and inclination
2) constrain all final orbital elements
3) option 2 with unconstrained true longitude

For option 1, the final orbit inclination is constrained by enforcing

$$\sqrt{h^2 + k^2} = \tan\left(\frac{i}{2}\right)$$

where *i* is the mission orbit inclination.

If the final orbit is circular, the code enforces the following two equality constraint $f = 0$ and $g = 0$.

Otherwise, for an elliptical mission orbit, the single equality constraint $\sqrt{f^2 + g^2} = e$ is enforced, where *e* is the park orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the user-defined final modified equinoctial orbital elements as follows

$$p_L = p_U = p_i \qquad f_L = f_U = f_i \qquad g_L = g_U = g_i$$
$$h_L = h_U = h_i \qquad k_L = k_U = k_i$$

Option 3 is identical to option 2 with the final true longitude unbounded.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the orbital transfer.

$$0.05 m_{sc_i} \le m_{sc} \le 1.05 m_{sc_i} \qquad 100\, p_f \le p \le 0.8\, p_i$$
$$-1 \le f \le +1 \qquad -1 \le g \le +1 \qquad -1 \le h \le +1 \qquad -1 \le k \le +1$$

where m_{sc_i} is the initial spacecraft mass. Finally, the three components of the unit thrust vector are constrained as follows

$$-1.1 \le u_r \le +1.1 \qquad -1.1 \le u_t \le +1.1 \qquad -1.1 \le u_n \le +1.1$$

Technical discussion

In this computer program, the orbital motion of the spacecraft is modeled in the modified equinoctial orbital elements coordinate system. Please consult *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for a definition of these orbital elements along with the equations of motion and other useful information about this system.

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)} \hat{\mathbf{u}}_T$$

where *T* is the thrust magnitude, *m* is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_r} & u_{T_t} & u_{T_n} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system.

The components of this unit vector are the control variables.

The propellant mass flow rate is determined from $\dot{m} = dm/dt = T/g\,I_{sp}$ where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g\,I_{sp}$ is also called the *exhaust velocity*. The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}\,t$ where m_{sc_i} is the initial mass of the spacecraft and $\dot{m}$ is the propellant flow rate.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

Finally, the pitch and yaw angles can be determined from the components of the unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the "local horizontal" and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four-quadrant operation.

The `oneburn_sos` software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems. Please consult *Appendix G – Aerospace Trajectory Coordinates and Time Systems* which summarizes the inertial and classical orbital elements-to/from-modified equinoctial coordinate relationships and transformations.

In the this computer program, the components of the inertial unit thrust vector are defined in terms of the right ascension α and the declination angle δ as follows

$$u_{T_{ECI_x}} = \cos\alpha\cos\delta \qquad u_{T_{ECI_y}} = \sin\alpha\cos\delta \qquad u_{T_{ECI_z}} = \sin\delta$$

Finally, the right ascension and declination angles can be determined from the components of the ECI unit thrust vector according to

$$\alpha = \tan^{-1}\left(u_{T_{ECI_y}}, u_{T_{ECI_x}}\right) \qquad \delta = \sin^{-1}\left(u_{T_{ECI_z}}\right)$$

where the calculation for right ascension is again a four-quadrant inverse tangent operation.

Example LEO-to-LEO orbit transfer

This section illustrates the orbit transfer for a single maneuver non-coplanar LEO-to-LEO example. For this example, all the orbital elements except true longitude for both the initial and final orbits are fixed. The main portion of the simulation definition file for this example is as follows

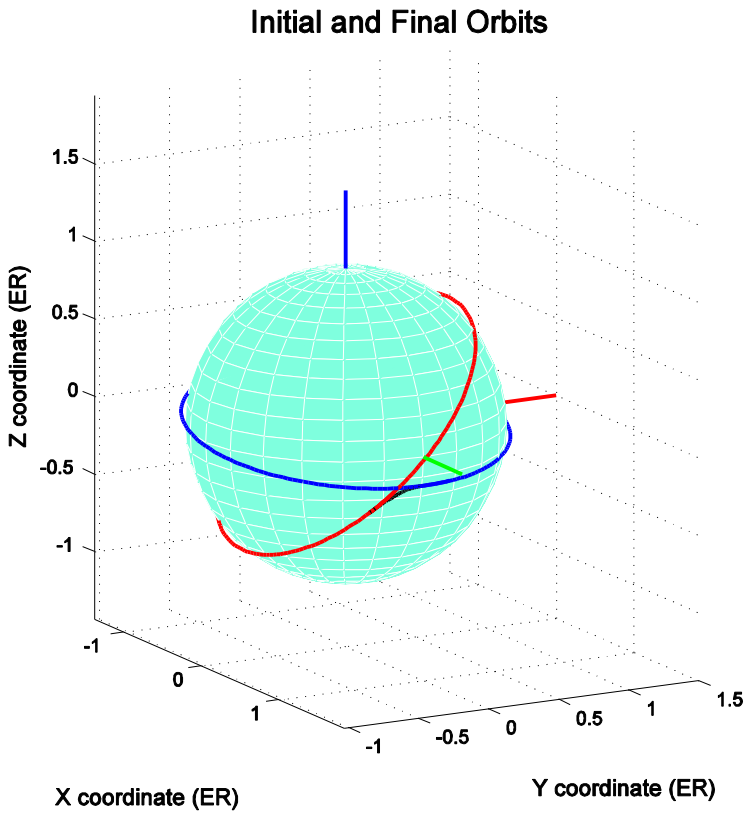
```
*********************************************
** earth-orbit trajectory optimization
** single finite-burn maneuver with final coast
** program oneburn_sos
** leo2leo.in - February 21, 2012
*********************************************

initial spacecraft mass (kilograms)
10000.0
```

```
thrust magnitude (newtons)
99200.0

specific impulse (seconds)
450.0

*******************************
type of propulsive initial guess
*******************************
1 = thrust duration
2 = delta-v
-------------------
2

initial guess for delta-v (meters/second)
5800.0d0

initial guess for thrust duration (seconds)
4550.0

lower bound for thrust duration (seconds)
0.01

upper bound for thrust duration (seconds)
10000.0

**************
coast maneuver
**************

initial guess for coast duration (seconds)
10.0

lower bound for coast duration (seconds)
1.0
upper bound for coast duration (seconds)
200.0

*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
6563.14

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
20.0d0

true anomaly (degrees)
0.0d0

*******************************
initial orbit constraint options
*******************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----------------------------------------------
3

***************
* FINAL ORBIT *
***************

semimajor axis (kilometers)
6728.14d0
```

```
orbital eccentricity (non-dimensional)
0.0d0

orbital inclination (degrees)
51.6d0

argument of perigee (degrees)
0.0d0

right ascension of the ascending node (degrees)
20.0d0

true anomaly (degrees)
0.0d0

*****************************
final orbit constraint options
*****************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all final orbital elements
3 = option 2 with unconstrained true longitude
----------------------------------------------
3

************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate gravity model
------------------------
2

************************
* initial guess options *
************************
 1 = numerical integration
 2 = binary data file
---------------------
1

name of binary initial guess data file
leo2leo.rsbin

*****************************
* binary restart file option *
*****************************

create/update binary restart file (yes or no)
no

*********************************************
* type of comma-delimited solution data file *
*********************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1

number of user-defined nodes or print step size in solution data file
100

name of solution output file
leo2leo.csv

*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
```

```
1

relative error in the objective function (performance index)
1.0d-5

relative error in the solution of the differential equations
1.0d-7

maximum number of mesh refinement iterations
20

maximum number of function evaluations
500000

maximum number of algorithm iterations
10000

**************************
sparse NLP iteration output
--------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2

**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1

***************************
differential equation output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The following is the 3-D graphics display for this example. The initial orbit trace is blue, the final orbit is red and the transfer maneuver trace is black.



Here are the numerical results created by the `oneburn_sos` computer program for this example.

```
program oneburn_sos
===================

input file ==> leo2leo.in

oblate earth gravity model

------------------------
beginning of finite burn
------------------------

mission elapsed time     00:00:00.000

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.656314000000D+04    0.231260711765D-15    0.285000000000D+02    0.000000000000D+00

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.200000000000D+02    0.349632240730D+03    0.349632240730D+03    0.881915957810D+02

     rx (km)                ry (km)               rz (km)               rmag (km)
 0.642165842152D+04    0.123266976185D+04    -.563591195034D+03    0.656314000000D+04

     vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 -.986248713166D+00    0.681033027729D+01    0.365785673126D+01    0.779315089527D+01

------------------
end of finite burn
------------------

mission elapsed time     00:04:19.701

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.672939043691D+04    0.346714306486D-03    0.516042165183D+02    0.315662746264D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.200014039317D+02    0.513704955945D+02    0.703324185873D+01    0.915636904293D+02
```

```
     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.609955227473D+04    0.276472241466D+04    0.645646678312D+03    0.672793338370D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 -.250687876646D+01    0.413739153123D+01    0.598798676484D+01    0.769795007313D+01

final mass              4162.16070871686      kilograms

propellant mass         5837.83929128314      kilograms

thrust duration         259.701018232233      seconds

delta-v                 3868.21300678244      meters/second

---------------------------
beginning of coast maneuver
---------------------------

mission elapsed time     00:04:19.701

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.672939043691D+04    0.346714306485D-03    0.516042165183D+02    0.315662746264D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.200014039317D+02    0.513704955945D+02    0.703324185873D+01    0.915636904293D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.609955227473D+04    0.276472241466D+04    0.645646678312D+03    0.672793338370D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 -.250687876646D+01    0.413739153123D+01    0.598798676484D+01    0.769795007313D+01

---------------------
end of coast maneuver
---------------------

mission elapsed time     00:07:39.701

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.672814000000D+04    0.351002915346D-15    0.516000000000D+02    0.000000000000D+00

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.200000000000D+02    0.201443655965D+02    0.201443655965D+02    0.915381704433D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.544337714471D+04    0.351284568192D+04    0.181588224703D+04    0.672814000000D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 -.402604621147D+01    0.331121293169D+01    0.566309181054D+01    0.769699863782D+01

coast duration          200.000000000023      seconds
                        3.33333333333372      minutes

=======================================
verification of optimal control solution
=======================================
------------------
end of finite burn
------------------

mission elapsed time     00:04:19.701

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.672939043691D+04    0.346714306486D-03    0.516042165183D+02    0.315662746264D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.200014039317D+02    0.513704955945D+02    0.703324185873D+01    0.915636904293D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.609955227473D+04    0.276472241466D+04    0.645646678312D+03    0.672793338370D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 -.250687876646D+01    0.413739153123D+01    0.598798676484D+01    0.769795007313D+01

final mass              4162.16070871692      kilograms
```

```
propellant mass         5837.83929128308      kilograms

thrust duration         259.701018232233      seconds

delta-v                 3868.21189633888      meters/second

final mission orbit
-------------------

mission elapsed time     00:07:39.701

      sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.672814024373D+04    0.364896455298D-07    0.515999998019D+02    0.490431457999D+01

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.199999999785D+02    0.152400513000D+02    0.201443658800D+02    0.915381754174D+02

      rx (km)                 ry (km)              rz (km)              rmag (km)
 0.544337713202D+04    0.351284570367D+04    0.181588226839D+04    0.672814000686D+04

      vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 -.402604625682D+01    0.331121302687D+01    0.566309190148D+01    0.769699876939D+01
```

Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `oneburn_sos` software.

The simulation summary screen display contains the following information

mission elapsed time = simulation time since beginning of maneuver

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (min) = orbital period in minutes

rx (km) = x-component of the spacecraft’s position vector in kilometers

ry (km) = y-component of the spacecraft’s position vector in kilometers

rz (km) = z-component of the spacecraft’s position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft’s position vector in kilometers

vx (km/sec) = x-component of the spacecraft’s velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft’s velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft’s velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft’s velocity vector in kilometers per second

final mass = final spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node or user-defined step size.

The user-defined comma-separated-variable (csv) disk file is created by the `odeprt` Fortran subroutine and contains the following information.

time (sec) = simulation time since ignition in seconds

time (min) = simulation time since ignition in minutes

semimajor axis (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

arg of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (min) = orbital period in minutes

mass (kg) = spacecraft mass in kilograms

thracc (mps/s) = thrust acceleration in meters/second**2

perigee altitude = perigee altitude in kilometers

apogee altitude = apogee altitude in kilometers

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

ut-eci-x = x-component of eci unit thrust vector

ut-eci-y = y-component of eci unit thrust vector

ut-eci-z = z-component of eci unit thrust vector

semi-parameter = orbital semiparameter in kilometers

f equinoctial element = modified equinoctial orbital element

g equinoctial element = modified equinoctial orbital element

h equinoctial element = modified equinoctial orbital element

k equinoctial element = modified equinoctial orbital element

true longitude = true longitude in degrees

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = magnitude of spacecraft's position vector in kilometers

vx (km) = x-component of the spacecraft's velocity vector in kilometers/second

vy (km) = y-component of the spacecraft's velocity vector in kilometers/second

vz (km) = z-component of the spacecraft's velocity vector in kilometers/second

vmag (km) = magnitude of spacecraft's velocity vector in kilometers/second

rasc (deg) = inertial right ascension of the unit thrust vector in degrees

decl (deg) = inertial declination of the unit thrust vector in degrees

yaw (deg) = out-of-plane yaw angle of the unit thrust vector in degrees

pitch (deg) = in-plane pitch angle of the unit thrust vector in degrees

fpa (deg) = inertial flight path angle in degrees

deltav (mps) = accumulative delta-v in meters per second

The `orbits.csv` file contains the following information.

time (seconds) = simulation time since ignition in seconds

rp1-x (er) = x-component of the initial orbit position vector in earth radii
rp1-y (er) = y-component of the initial orbit position vector in earth radii
rp1-z (er) = z-component of the initial orbit position vector in earth radii
rp2-x (er) = x-component of the final orbit position vector in earth radii
rp2-y (er) = y-component of the final orbit position vector in earth radii
rp2-z (er) = z-component of the final orbit position vector in earth radii

"Direct Trajectory Optimization Using Nonlinear Programming and Collocation", C. R. Hargraves and S. W. Paris, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 10, No. 4, July-August, 1987, pp. 338-342.

"Optimal Finite-Thrust Spacecraft Trajectories Using Direct Transcription and Nonlinear Programming", Paul J. Enright, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1991.

"Improved Collocation Methods with Application to Direct Trajectory Optimization", Albert L. Herman, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1995.

"Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

CMATO 2 – Two Maneuver, Finite-Burn Trajectory Optimization

This *CMATO* application is a Fortran computer program named `twoburn_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the classic orbit transfer trajectory optimization problem. The software models the trajectory as a three phase mission in the sequence burn-coast-burn. The two burns are simulated as constant-thrust, finite-burn propulsive maneuvers. This computer program attempts to maximize the spacecraft mass at the end of the final propulsive maneuver.

The important features of this scientific simulation are

- two finite-burn, continuous thrust orbital maneuvers
- variable attitude steering
- constant propulsive thrust magnitude and specific impulse
- modified equinoctial equations of motion with oblate Earth gravity model
- user-specified initial and final orbit constraints

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `twoburn_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method. The `twoburn_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `twoburn_sos` software is "data-driven" by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in times font. This example attempts to optimize the maneuvers required to transfer a spacecraft from a near circular low Earth orbit (LEO) to a typical geosynchronous Earth orbit (GEO).

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
**************************************
** two maneuver, finite-burn earth-orbit
** trajectory optimization
** program twoburn_sos
** leo2geo.in - April 15, 2012
**************************************
```

The first input is the initial mass of the entire spacecraft in kilograms.

```
initial spacecraft mass (kilograms)
15000.0
```

This next integer input defines the type of initial guess for the propulsive maneuver.

```
*******************************
type of propulsive initial guess
*******************************
1 = thrust duration
2 = delta-v magnitude
---------------------
2
```

The next four inputs define the thrust magnitude and the specific impulse of the upper stage or spacecraft propulsion system, and the user's initial guess for either the delta-v or thrust duration for the first maneuver.

```
-------------------------
first propulsive maneuver
-------------------------
thrust magnitude (newtons)
25000.0

specific impulse (seconds)
400.0

initial guess for delta-v (meters/second)
2480.0

initial guess for thrust duration (seconds)
170.0
```

The next four inputs define the thrust magnitude and the specific impulse of the upper stage or spacecraft propulsion system, and the user's initial guess for either the delta-v or thrust duration for the second maneuver.

```
-------------------------
second propulsive maneuver
-------------------------

thrust magnitude (newtons)
5000.0

specific impulse (seconds)
350.0

initial guess for delta-v (meters/second)
1790.0

initial guess for thrust duration (seconds)
170.0
```

The next three inputs define the user's initial guess for the duration of the coast phase along with a lower and upper bound for the coast duration.

```
-----------
coast phase
-----------

initial guess for coast duration (minutes)
315.0

lower bound for coast duration (minutes)
200.0

upper bound for coast duration (minutes)
400.0
```

The next six inputs define the classical orbital elements of the initial park orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
6563.14

orbital eccentricity (non-dimensional)
0.015

orbital inclination (degrees)
28.5

argument of perigee (degrees)
120.0

right ascension of the ascending node (degrees)
100.0

true anomaly (degrees)
0.0
```

This next integer input allows the user to define the type of initial orbit constraints to use during the simulation. Please see the "Problem setup" section later in this document for information about this program option.

```
******************************
initial orbit constraint options
******************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
---------------------------------------------
1
```

The next six inputs define the classical orbital elements of the final mission orbit. These elements are also defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
**************
* FINAL ORBIT *
**************

semimajor axis (kilometers)
42166.263

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
2.5

argument of perigee (degrees)
300.0

right ascension of the ascending node (degrees)
120.0

true anomaly (degrees)
0.0
```

This next integer input allows the user to define the type of final orbit constraints to use during the simulation. Please see the "Problem setup" section later in this document for information about this program option.

```
*****************************
final orbit constraint options
*****************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all final orbital elements
3 = option 2 with unconstrained true longitude
---------------------------------------------
3
```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will use a J_2 gravity model in the spacecraft equations of motion.

```
************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate gravity model
------------------------
2
```

This next input defines the type of initial guess to use. Please see the technical discussion section for information about how the first option is modeled. Option 2 requires either a binary restart file created from a previous run using either initial guess option 1 or an updated binary restart file. This feature is described in the next two sections.

```
************************
* initial guess options *
************************
 1 = numerical integration
 2 = binary data file
---------------------
1
```

If the user elects to use a binary data file (option 2 above) for the initial guess, the following text input specifies the name of the file to use.

```
name of binary initial guess data file
leo2geo.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess option 1, the software will create a binary restart file. For initial guess option 2, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*****************************
* binary restart file option *
*****************************

create/update binary data file (yes or no)
no
```

This next input specifies the type of solution data file to create.

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = OC-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
25
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
leo2geo.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
******************************
* algorithm control parameters *
******************************
```

```
discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
10000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the *Sparse Optimization Suite* NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the *Sparse Optimization Suite* optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the Sparse Optimization Suite differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
**************************
differential equation output
--------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string no.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of inputs allow the reading and writing of configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `twoburn_sos` software. Please consult *Appendix J – Sparse Optimization Suite Configuration File*.

```
************************************
* optimal control configuration options
************************************

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
leo2geo_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
leo2geo_config1.txt
```

Optimal control solution

The following is the `twoburn_sos` solution for this example. The output includes the time and orbital characteristics at the beginning and end of each mission phase.

```
 program twoburn_sos
 ===================

 input file ==> leo2geo.in

 numerical integration initial guess

 oblate earth gravity model

 -------------------------------------
 beginning of first propulsive maneuver
 -------------------------------------

 mission elapsed time     00:00:00.000
```

```
     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.656314000000D+04    0.150000000004D-01    0.285000000000D+02    0.179196327908D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.119947466863D+03    0.318037646103D+03    0.137233974011D+03    0.881916022527D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.977069953510D+03    -.606099670550D+04    0.210248702383D+04    0.648928335362D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.708880451413D+01    -.202549730354D+01    -.278600998827D+01    0.788134762721D+01

--------------------------------
end of first propulsive maneuver
--------------------------------

mission elapsed time     00:18:38.659

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.244711724047D+05    0.725948454612D+00    0.264721389995D+02    0.179895113440D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.119828797416D+03    0.407077273961D+02    0.220602840836D+03    0.634953461869D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.659297946572D+04    -.275407175855D+04    -.216599500646D+04    0.746617866480D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.487228379563D+01    0.718704821378D+01    -.388504453231D+01    0.951243304473D+01
```

The following program output is the spacecraft mass, the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v for the first maneuver.

```
spacecraft mass           7870.53129355029      kilograms

propellant mass           7129.46870644971      kilograms

thrust duration           1118.65926864158      seconds
                          18.6443211440263      minutes

delta-v                   2529.82034928729      meters/second
```

This section of the numeric results summarizes the time and orbital conditions at the beginning and end of the transfer orbit coast.

```
------------------------
beginning of coast phase
------------------------

mission elapsed time     00:18:38.659

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.244711724047D+05    0.725948454612D+00    0.264721389995D+02    0.179895113440D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.119828797416D+03    0.407077273961D+02    0.220602840836D+03    0.634953461869D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.659297946572D+04    -.275407175855D+04    -.216599500646D+04    0.746617866480D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.487228379563D+01    0.718704821379D+01    -.388504453231D+01    0.951243304473D+01

------------------
end of coast phase
------------------
```

```
mission elapsed time     05:07:30.284

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.244242477038D+05    0.725339679047D+00    0.264690756581D+02    0.179961581757D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.119758219971D+03    0.177464376454D+03    0.357425958210D+03    0.633128004987D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.193739413380D+05    0.372905472704D+05    -.841352842717D+03    0.420314452916D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.137738985018D+01    -.487203868378D+00    0.715777169774D+00    0.162693188718D+01

coast duration            17331.6249596538      seconds
                          288.860415994230      minutes
                          4.81434026657050      hours

---------------------------------------
beginning of second propulsive maneuver
---------------------------------------

mission elapsed time     05:07:30.284

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.244242477038D+05    0.725339679047D+00    0.264690756581D+02    0.179961581757D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.119758219971D+03    0.177464376454D+03    0.357425958210D+03    0.633128004987D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.193739413380D+05    0.372905472704D+05    -.841352842717D+03    0.420314452916D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.137738985018D+01    -.487203868378D+00    0.715777169774D+00    0.162693188718D+01

---------------------------------
end of second propulsive maneuver
---------------------------------

mission elapsed time     05:43:11.886

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.421662630000D+05    0.157018823694D-15    0.250000000000D+01    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.120000000000D+03    0.383296455222D+01    0.383296455222D+01    0.143617512421D+04

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.234747395985D+05    0.350273495880D+05    0.122951225622D+03    0.421662630000D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.255141829470D+01    -.171038725638D+01    0.133811476571D+00    0.307458377550D+01
```

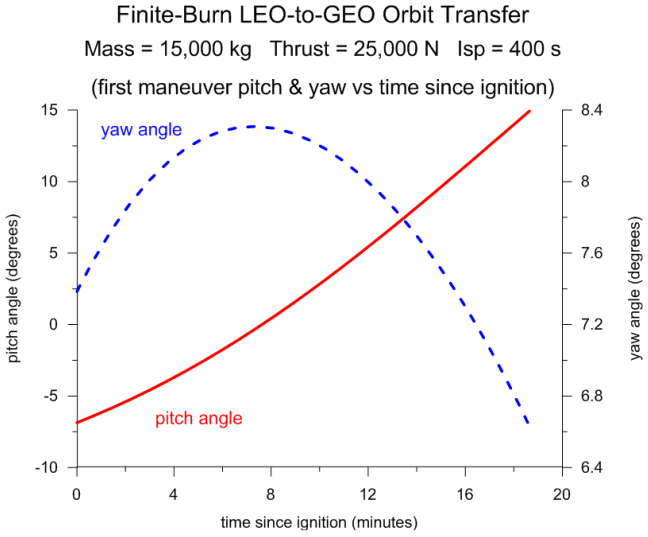
The following program output is the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v for the second maneuver.

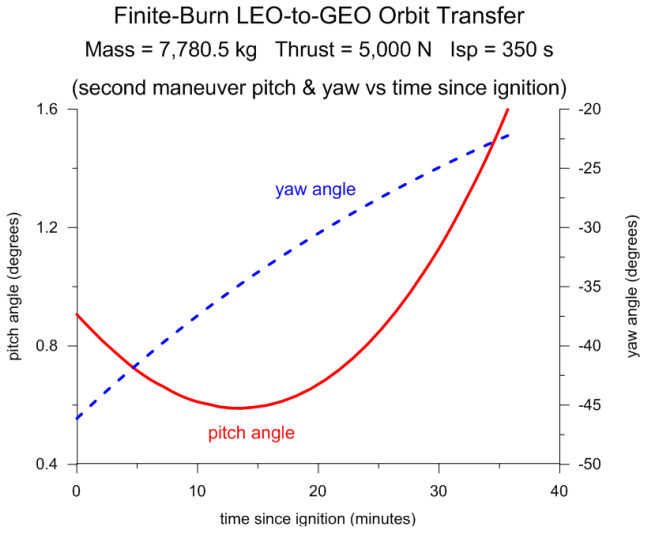
```
propellant mass           3119.75087432484      kilograms

thrust duration           2141.60134383684      seconds
                          35.6933557306139      minutes

delta-v                   1732.69627276496      meters/second
```

After the simulation is complete, the software will display a simulation summary.

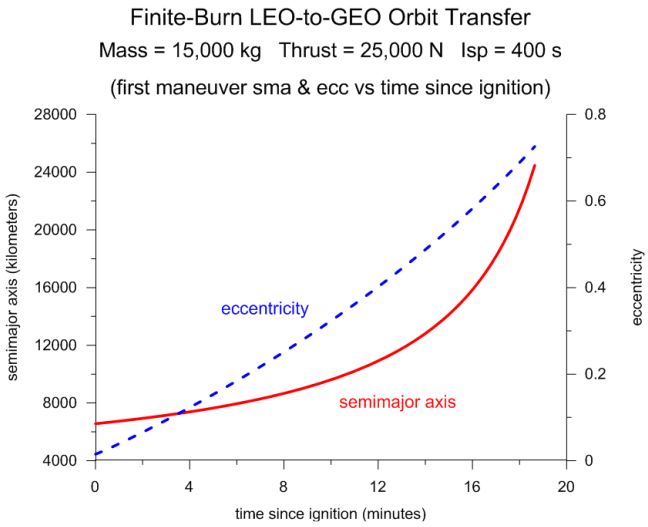
```
SIMULATION SUMMARY
------------------

initial spacecraft mass    15000.0000000000      kilograms

total propellant mass      10249.2195807745      kilograms

final spacecraft mass      4750.78041922545      kilograms


total delta-v              4262.51662205225      meters/second

total thrust duration      3260.26061247841      seconds
                           54.3376768746402      minutes
```

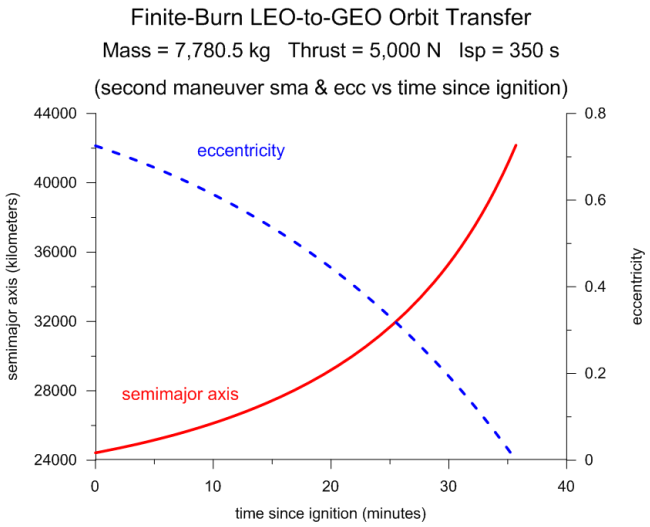
The following two plots illustrate the evolution of the pitch and yaw steering angles during the first and second finite-burn maneuver.



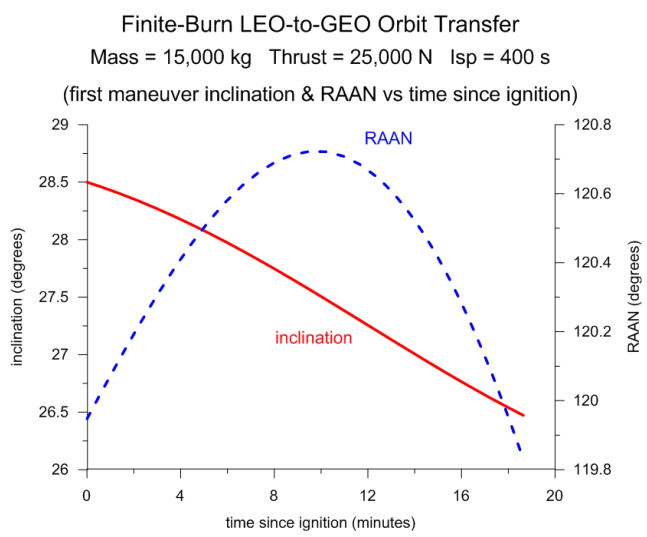


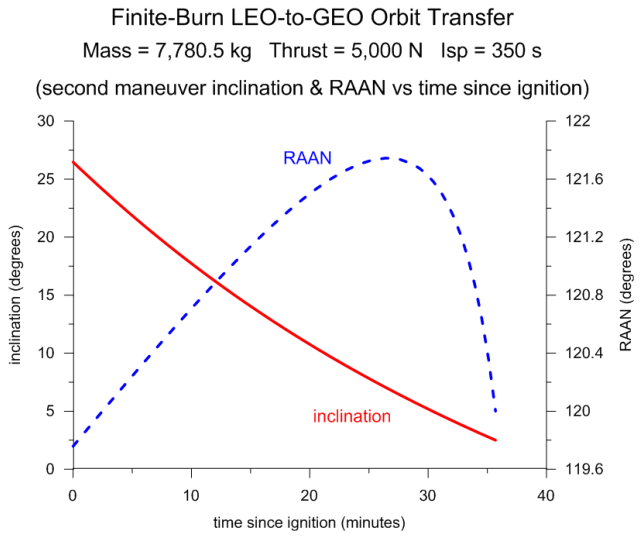
The next pair of plots illustrate the behavior of the semimajor axis and orbital eccentricity during the first and second maneuvers.



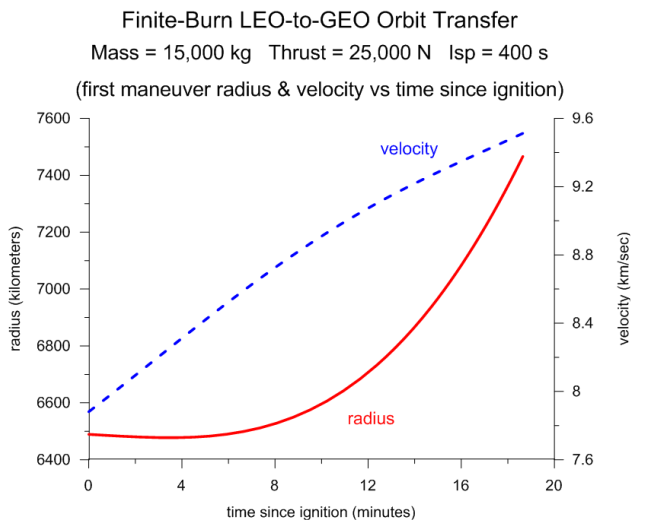


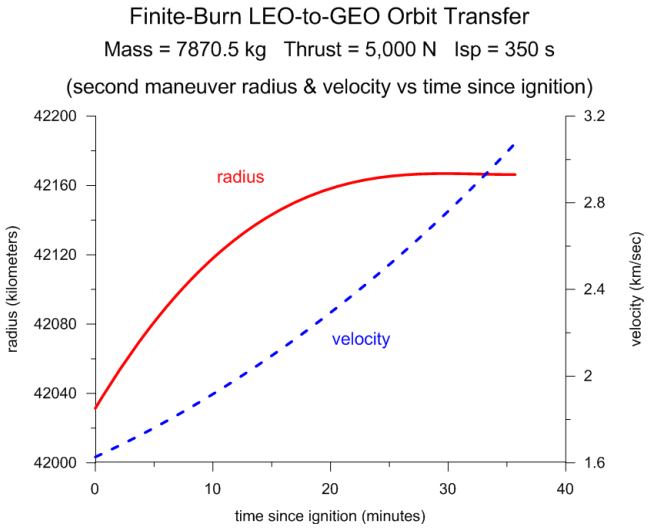
The next pair of plots illustrate the behavior of the orbital inclination and right ascension of the ascending node (RAAN) during the first and second maneuvers.





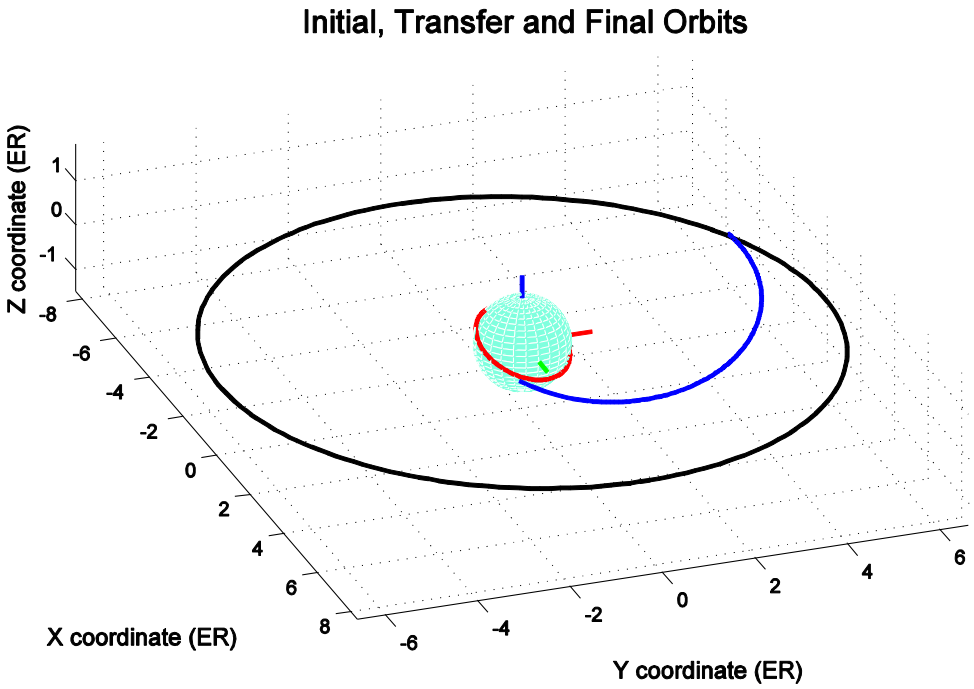
The following two plots illustrate the evolution of the geocentric radius and velocity during the each finite-burn maneuver.





The `twoburn_sos` computer program will also create three output files named `orbit1.csv`, `orbit2.csv` and `orbit3.csv`. This file contains the Earth-centered inertial position vectors of the park, transfer and final mission orbit. The `twoburn_sos` software package includes a MATLAB script named `oplot.m` that can be used to create trajectory graphic displays using these data files. The interactive graphic features of MATLAB allow the user to rotate and zoom the displays. These capabilities allow the user to interactively find the best viewpoint as well as verify basic three-dimensional geometry of the orbital transfer.

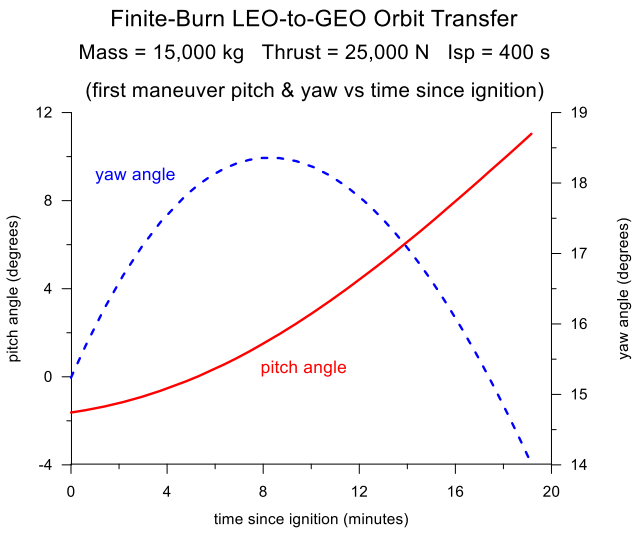
The following is the graphics display for this example. The initial orbit trace is red, the transfer orbit is blue and the final mission orbit is black. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis.

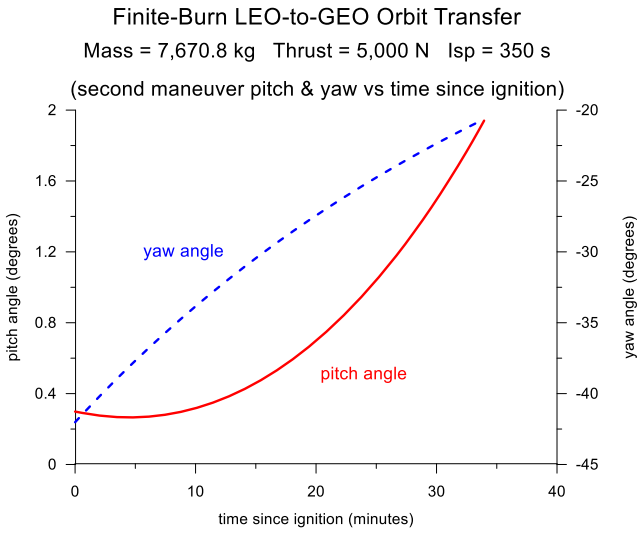


Solid propellant upper stage

Suppose the first maneuver of the orbit transfer is performed with a solid propellant stage. This type of propulsion has a fixed thrust duration. The propellant in the upper stage is designed to provide a delta-v slightly higher than the required first maneuver delta-v to account for performance dispersions. To implement this "fixed delta-v" simulation in the *SOS* software, the thrust duration of the first maneuver is constrained and no longer determined by the software.

The following two plots illustrate the pitch and yaw behavior for both maneuvers for the previous simulation example. The only change to the simulation was constraining the first thrust duration to 1150 seconds. In the original simulation, the duration was 1118 seconds. The longer thrust duration produces an "excess" delta-v of 100 meters/second.





Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* software can be verified by numerically integrating the orbital equations of motion with the optimal control solution and the initial park orbit conditions determined by the software. This is equivalent to solving an initial value problem (IVP) that uses the optimal unit thrust vector solution.

This part of the `twoburn_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is a display of the final solution computed using this *explicit* numerical integration method.

```
=======================================
verification of optimal control solution
=======================================

------------------
final mission orbit
------------------

mission elapsed time     05:43:11.886

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.421662635019D+05    0.160175963159D-07    0.250000121139D+01    0.127337790391D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.119999993671D+03    0.351099192462D+03    0.383297150142D+01    0.143617514985D+04

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.234747398791D+05    0.350273491999D+05    0.122951507257D+03    0.421662628346D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.255141829452D+01    -.171038730613D+01    0.133811541581D+00    0.307458380586D+01

right ascension          123.829327888839      degrees

declination             0.167067507520765      degrees

final spacecraft mass    4750.78041919988      kilograms

first delta-v            2529.82005790686      meters/second

second delta-v           1732.69612356106      meters/second

total delta-v            4262.51618146792      meters/second
```

Creating an initial guess

The software allows the user to input either a delta-v or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. An estimate of the thrust duration can be determined from the following expression

$$t_d = \frac{I_{sp} m_p g}{F} = \frac{m_p V_{ex}}{F}$$

The propellant mass required for a given ΔV is a function of the initial (or final) mass of the spacecraft and the exhaust velocity as follows

$$m_p = m_i\left(1 - e^{\frac{-\Delta V}{V_{ex}}}\right) = m_f\left(e^{\frac{\Delta V}{V_{ex}}} - 1\right)$$

In these equations

$m_i =$ initial mass

$m_f =$ final mass

$m_p =$ propellant mass

$V_{ex} =$ exhaust velocity = $g I_{sp}$

$I_{sp} =$ specific impulse

$\Delta V =$ impulsive velocity increment

$F =$ thrust

$g =$ acceleration of gravity

For the thrust duration initial guess option, the software requires an initial guess for the thrust duration for each propulsive maneuver. All these inputs should be in seconds. If the `twoburn_sos` computer program cannot find a feasible solution, try increasing the guess for thrust duration.

The software uses a tangential thrusting steering method to generate an initial guess for the optimal trajectory. For tangential thrusting, the unit thrust vector in the modified equinoctial frame at all times is simply $\mathbf{u}_T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$. Please note that this type of steering method creates a *coplanar* initial guess.

The dynamic variables and control variables at each grid point are determined by the *Sparse Optimization Suite* by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 4`. These program options create an initial guess from the numerical integration of the equations programmed in the `oderhs` subroutine. The number and location of the initial collocation nodes are determined from the variable step-size numerical integration.

Problem setup

This section provides additional details about the software implementation. It explains such things as point and path constraints, the performance index and the numerical technique used to create an initial guess for the software.

Point functions – initial orbit constraints

The software allows the user to select one of the following initial orbit constraint options:

1) constrain semimajor axis, eccentricity and inclination
2) constrain all initial orbital elements
3) option 2 with unconstrained true longitude

For option 1, the initial orbit inclination is constrained by enforcing $\sqrt{h^2 + k^2} = \tan\left(\frac{i}{2}\right)$ where i is the park orbit inclination.

If the park orbit is circular, the software enforces the following two equality constraints

$$f = 0 \text{ and } g = 0$$

Otherwise, for an elliptical park orbit, the single equality constraint

$$\sqrt{f^2 + g^2} = e$$

is enforced, where *e* is the user-defined park orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the initial modified equinoctial orbital elements as follows:

$$p_L = p_U = p_i \qquad f_L = f_U = f_i$$

$$g_L = g_U = g_i \qquad h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

Option 3 is identical to option 2 with the initial true longitude unbounded.

In optimal control terminology, these derived constraints or boundary conditions are called *point functions*.

Performance index – maximize final spacecraft mass

The objective function or performance index *J* for this simulation is the mass of the spacecraft at burnout or termination of the propulsive maneuver. This is simply $J = m_f$.

The value of the `maxmin` indicator in the *Sparse Optimization Suite* algorithm tells the software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the user-defined initial value.

Path constraint – unit thrust vector scalar magnitude

For the *variable steering* program option, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as follows:

$$\left|\mathbf{u}_T\right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

Point functions – final mission orbit constraints

The software allows the user to select one of the following final orbit constraint options:

1) constrain semimajor axis, eccentricity and inclination
2) constrain all final orbital elements
3) option 2 with unconstrained true longitude

For option 1, the final orbital inclination is constrained by enforcing $\sqrt{h^2+k^2} = \tan(i/2)$ where *i* is the user-defined mission orbit inclination.

If the final orbit is circular, the software enforces the following two equality constraints

$$f = 0 \text{ and } g = 0$$

Otherwise, for an elliptical mission orbit, the single equality constraint

$$\sqrt{f^2+g^2} = e$$

is enforced, where *e* is the user-defined mission orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the user-defined final modified equinoctial orbital elements according to

$$p_L = p_U = p_i \qquad f_L = f_U = f_i$$
$$g_L = g_U = g_i \qquad h_L = h_U = h_i$$
$$k_L = k_U = k_i$$

Option 3 is identical to option 2 with the final true longitude unbounded.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the orbital transfer.

$$0.05m_{sc_i} \le m_{sc} \le 1.05m_{sc_i} \qquad 100\,p_f \le p \le 0.8\,p_i$$
$$-1 \le f \le +1 \qquad -1 \le g \le +1$$
$$-1 \le h \le +1 \qquad -1 \le k \le +1$$

where m_{sc_i} is the initial spacecraft mass.

Finally, the three components of the unit thrust vector are constrained as

$$-1.1 \le u_r \le +1.1 \qquad -1.1 \le u_t \le +1.1 \qquad -1.1 \le u_n \le +1.1$$

Technical discussion

In this computer program, the orbital motion of the spacecraft is modeled in the modified equinoctial orbital elements coordinate system. Please consult *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for a definition of these orbital elements along with the equations of motion in this system.

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)}\hat{\mathbf{u}}_T$$

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_r} & u_{T_t} & u_{T_n} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system. *The components of this unit vector are the control variables.*

The propellant mass flow rate is determined from

$$\dot{m} = \frac{dm}{dt} = \frac{T}{g\, I_{sp}}$$

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g\, I_{sp}$ is also called the *exhaust velocity*.

The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}\, t$ where m_{sc_i} is the initial mass of the spacecraft and $\dot{m}$ is the propellant flow rate.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as follows

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

Finally, the pitch and yaw angles can be determined from the components of the unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the "local horizontal" and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four-quadrant operation.

The `twoburn_sos` software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems.

Example LEO-to-ISS orbit transfer

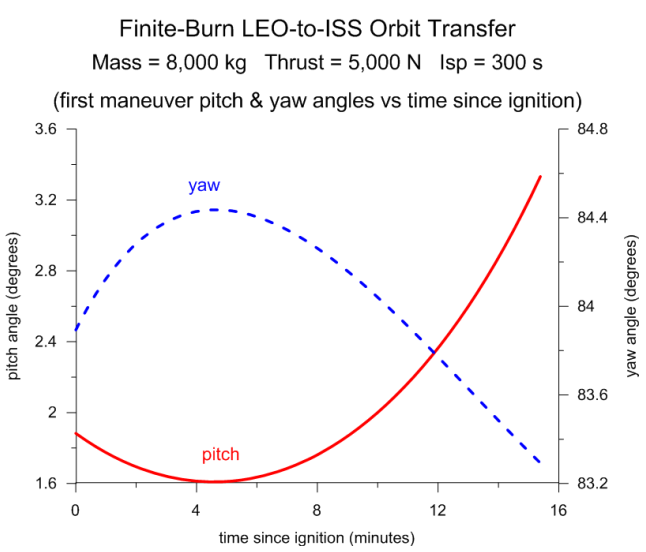
This section illustrates the `twoburn_sos` solution and trajectory graphics for a variable attitude, medium-thrust LEO-to-ISS (International Space Station) orbit transfer. For this example, impulsive delta-v guesses were determined using software that calculates an impulsive Hohmann transfer solution for this problem.

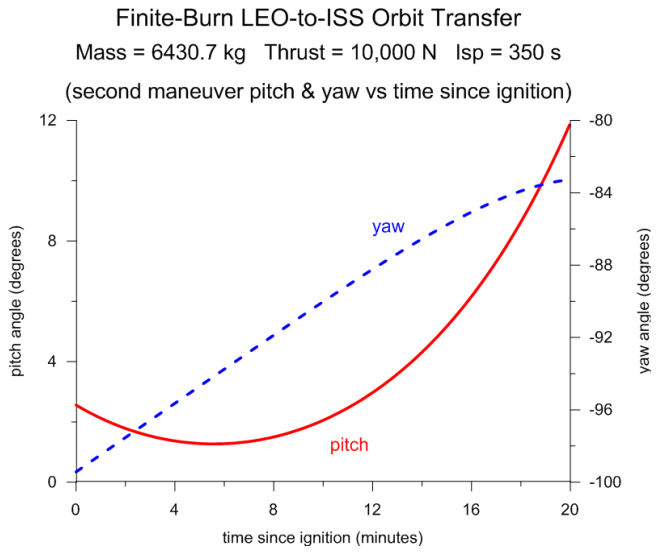
The following is the first part of the input data file for this example.

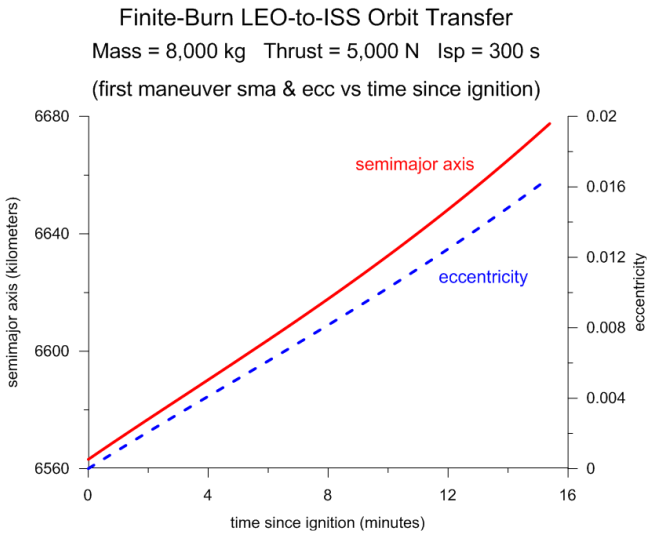
```
**************************************
** two maneuver, finite-burn earth-orbit
** trajectory optimization
** program twoburn_sos
** leo2iss.in - April 16, 2012
**************************************

initial spacecraft mass (kilograms)
8000.0

**************************************
type of propulsive maneuver initial guess
**************************************
1 = thrust duration
2 = delta-v magnitude
---------------------
2

-------------------------
first propulsive maneuver
-------------------------

thrust magnitude (newtons)
5000.0

specific impulse (seconds)
300.0

initial guess for delta-v (meters/second)
180.0

initial guess for thrust duration (seconds)
170.0

-------------------------
second propulsive maneuver
-------------------------
thrust magnitude (newtons)
10000.0

specific impulse (seconds)
350.0

initial guess for delta-v (meters/second)
2905.0

initial guess for thrust duration (seconds)
80.0

-----------
coast phase
-----------

initial guess for coast duration (minutes)
45.0

lower bound for coast duration (minutes)
20.0

upper bound for coast duration (minutes)
60.0

*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
6563.14

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5
```

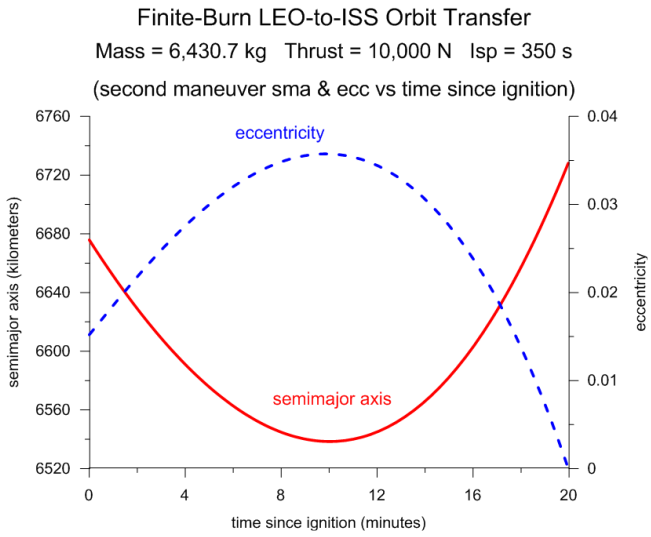
```
argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
100.0

true anomaly (degrees)
0.0

*******************************
initial orbit constraint options
*******************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----------------------------------------------
1

***************
* FINAL ORBIT *
***************

semimajor axis (kilometers)
6728.14

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
51.6

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
100.0

true anomaly (degrees)
0.0


*****************************
final orbit constraint options
*****************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all final orbital elements
3 = option 2 with unconstrained true longitude
-----------------------------------------------
3

************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate gravity model
------------------------
2
```

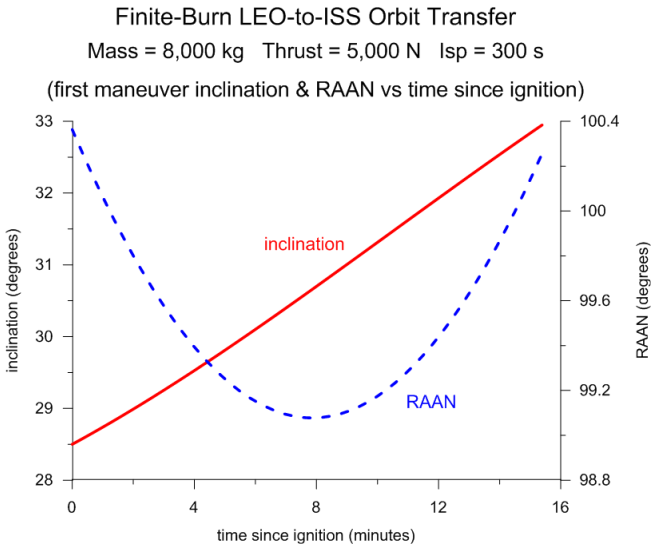
The following are plots of the steering and trajectory characteristics for this example.

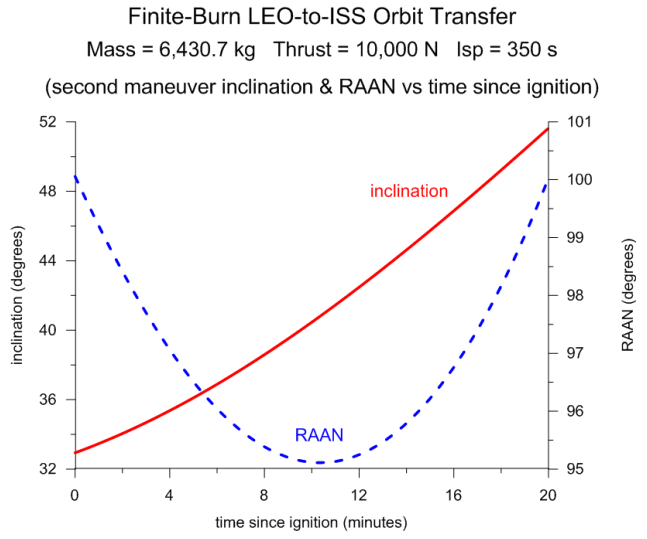

Finite-Burn LEO-to-ISS Orbit Transfer
Mass = 8,000 kg Thrust = 5,000 N Isp = 300 s
(first maneuver pitch & yaw angles vs time since ignition)
yaw
pitch
pitch angle (degrees)
yaw angle (degrees)
time since ignition (minutes)

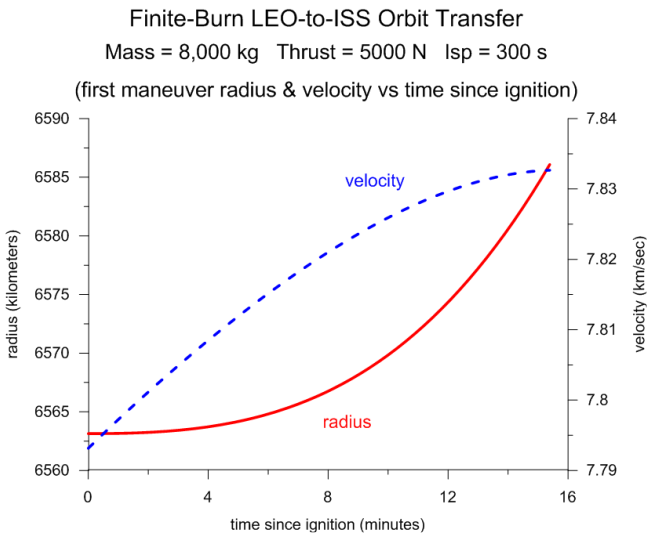

Finite-Burn LEO-to-ISS Orbit Transfer
Mass = 6430.7 kg Thrust = 10,000 N Isp = 350 s
(second maneuver pitch & yaw vs time since ignition)
yaw
pitch
pitch angle (degrees)
yaw angle (degrees)
time since ignition (minutes)

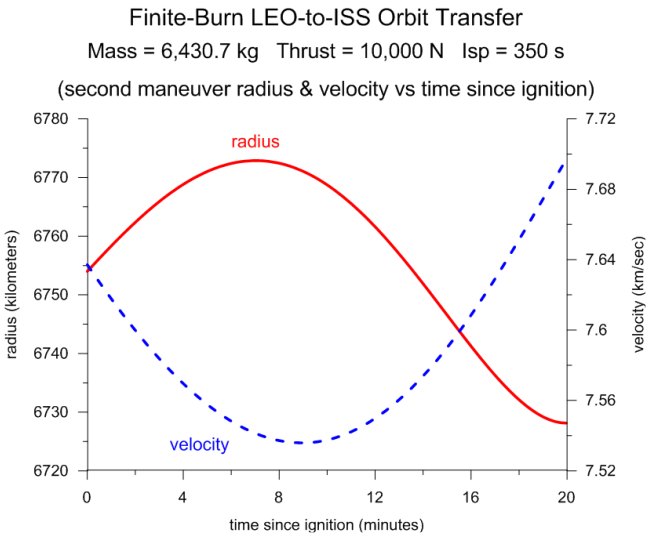

Finite-Burn LEO-to-ISS Orbit Transfer
Mass = 8,000 kg Thrust = 5,000 N Isp = 300 s
(first maneuver sma & ecc vs time since ignition)
semimajor axis
eccentricity
semimajor axis (kilometers)
eccentricity
time since ignition (minutes)


Finite-Burn LEO-to-ISS Orbit Transfer
Mass = 6,430.7 kg Thrust = 10,000 N Isp = 350 s
(second maneuver sma & ecc vs time since ignition)
eccentricity
semimajor axis
semimajor axis (kilometers)
eccentricity
time since ignition (minutes)


Finite-Burn LEO-to-ISS Orbit Transfer
Mass = 8,000 kg Thrust = 5,000 N Isp = 300 s
(first maneuver inclination & RAAN vs time since ignition)
inclination
RAAN
inclination (degrees)
RAAN (degrees)
time since ignition (minutes)


Finite-Burn LEO-to-ISS Orbit Transfer
Mass = 6,430.7 kg Thrust = 10,000 N Isp = 350 s
(second maneuver inclination & RAAN vs time since ignition)
inclination
RAAN
inclination (degrees)
RAAN (degrees)
time since ignition (minutes)





Here's the main program output and verification for this example.

```
program twoburn_sos
===================

input file ==> leo2iss.in

numerical integration initial guess

oblate earth gravity model

--------------------------------------
beginning of first propulsive maneuver
--------------------------------------

mission elapsed time      00:00:00.000

     sma (km)                eccentricity         inclination (deg)        argper (deg)
 0.656314000000D+04     0.110612553947D-15     0.285000000000D+02     0.000000000000D+00

    raan (deg)           true anomaly (deg)        arglat (deg)           period (min)
 0.100363294121D+03     0.326736552541D+03     0.326736552541D+03     0.881916022527D+02

     rx (km)                  ry (km)                rz (km)                rmag (km)
 0.212477171977D+04     0.596738793323D+04     -.171768246564D+04     0.656314000000D+04

     vx (kps)                 vy (kps)               vz (kps)               vmag (kps)
 -.640214595143D+01     0.317457253705D+01     0.310930995346D+01     0.779315032339D+01

--------------------------------
end of first propulsive maneuver
--------------------------------

mission elapsed time      00:15:23.359

     sma (km)                eccentricity         inclination (deg)        argper (deg)
 0.667754855184D+04     0.165328254596D-01     0.329474801412D+02     0.355233433521D+03

    raan (deg)           true anomaly (deg)        arglat (deg)           period (min)
 0.100255885100D+03     0.345888894857D+02     0.298223230068D+02     0.905076548286D+02

     rx (km)                  ry (km)                rz (km)                rmag (km)
 -.372197737672D+04     0.513323822355D+04     0.178135829385D+04     0.658608287403D+04

     vx (kps)                 vy (kps)               vz (kps)               vmag (kps)
 -.495861875658D+01     -.479160585935D+01     0.371527030388D+01     0.783266366668D+01

spacecraft mass            6430.72714842286      kilograms

propellant mass            1569.27285157714      kilograms
```

```
thrust duration           923.358576595192      seconds
                          15.3893096099199      minutes

delta-v                   642.396161808662      meters/second

------------------------
beginning of coast phase
------------------------

mission elapsed time     00:15:23.359

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.667754855184D+04    0.165328254596D-01    0.329474801412D+02    0.355233433521D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.100255885100D+03    0.345888894857D+02    0.298223230068D+02    0.905076548286D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.372197737672D+04    0.513323822355D+04    0.178135829385D+04    0.658608287403D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 -.495861875657D+01    -.479160585935D+01    0.371527030388D+01    0.783266366668D+01

------------------
end of coast phase
------------------

mission elapsed time     00:42:01.551

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.667584555215D+04    0.152066565657D-01    0.329380649128D+02    0.355274673206D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.100057403600D+03    0.140887797298D+03    0.136162470504D+03    0.904730332878D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.301488941672D+04    -.548245175517D+04    0.254353639062D+04    0.675399236106D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.544260814774D+01    -.446092094692D+01    -.296721491969D+01    0.763715676965D+01

coast duration            1598.19246759981      seconds
                          26.6365411266635      minutes
                         0.443942352111059      hours

---------------------------------------
beginning of second propulsive maneuver
---------------------------------------

mission elapsed time     00:42:01.551

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.667584555215D+04    0.152066565657D-01    0.329380649128D+02    0.355274673206D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.100057403600D+03    0.140887797298D+03    0.136162470504D+03    0.904730332878D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.301488941672D+04    -.548245175517D+04    0.254353639062D+04    0.675399236106D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.544260814774D+01    -.446092094692D+01    -.296721491969D+01    0.763715676965D+01

---------------------------------
end of second propulsive maneuver
---------------------------------

mission elapsed time     01:02:59.702

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.672814000000D+04    0.135806524537D-15    0.516000000000D+02    0.000000000000D+00

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.100000000000D+03    0.213965850265D+03    0.213965850265D+03    0.915381771606D+02
```

```
      rx (km)                 ry (km)              rz (km)              rmag (km)
 0.326840206482D+04    -.508989721860D+04    -.294590599076D+04    0.672814000000D+04

      vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.315821923111D+01    0.492352099988D+01    -.500283635428D+01    0.769699807300D+01

propellant mass           3490.78368184216      kilograms

thrust duration           1198.15128277386      seconds
                          19.9691880462309      minutes

delta-v                   2686.47371008881      meters/second

SIMULATION SUMMARY
------------------

initial spacecraft mass   8000.00000000000      kilograms

total propellant mass     5060.05653341929      kilograms

final spacecraft mass     2939.94346658071      kilograms

total delta-v             3328.86987189747      meters/second

total thrust duration     2121.50985936905      seconds
                          35.3584976561508      minutes

========================================
verification of optimal control solution
========================================

-------------------
final mission orbit
-------------------

mission elapsed time     01:02:59.702

      sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.672813998828D+04    0.600936604302D-08    0.516000000807D+02    0.204587101726D+02

     raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.999999996116D+02    0.193507139109D+03    0.213965849282D+03    0.915381769214D+02

      rx (km)                 ry (km)              rz (km)              rmag (km)
 0.326840199228D+04    -.508989733617D+04    -.294590593112D+04    0.672814002759D+04

      vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.315821930073D+01    0.492352086106D+01    -.500283638807D+01    0.769699803473D+01

right ascension          302.705946066975      degrees

declination             -25.9666865187013      degrees

final spacecraft mass    2939.94346658047      kilograms

first delta-v             642.396149630716      meters/second

second delta-v           2686.47335229412      meters/second

total delta-v            3328.86950192484      meters/second
```

Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `twoburn_sos` software.

The simulation summary screen display contains the following information:

mission elapsed time = simulation time since the beginning of the simulation

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (min) = orbital period in minutes

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

spacecraft mass = current spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node.

The comma-separated-variable disk file is created by the `odeprt` subroutine and contains the following information

time (sec) = mission elapsed time in seconds

time (min) = mission elapsed time in minutes

semimajor axis (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argument of perigee = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (min) = orbital period in minutes

mass (kg) = spacecraft mass in kilograms

thracc (mps/s) = thrust acceleration in meters/second**2

yaw (deg) = thrust vector yaw angle in degrees

pitch (deg) = thrust vector pitch angle in degrees

rasc (deg) = inertial right ascension in degrees

decl (deg) = inertial declination in degrees

perigee altitude = perigee altitude in kilometers

apogee altitude = apogee altitude in kilometers

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

semi-parameter = orbital semiparameter in kilometers

f equinoctial element = modified equinoctial orbital element

g equinoctial element = modified equinoctial orbital element

h equinoctial element = modified equinoctial orbital element

k equinoctial element = modified equinoctial orbital element

true longitude = true longitude in degrees

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

fpa (deg) = flight path angle in degrees

rmag (km) = geocentric radius in kilometers

vmag (kps) = velocity in kilometers per second

deltav1 (mps) = first maneuver accumulative delta-v in meters per second

deltav2 (mps) = second maneuver accumulative delta-v in meters per second

dvacc (mps) = total accumulative delta-v in meters per second

"Survey of Numerical Methods for Trajectory Optimization", John T. Betts, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998, pp. 193-207.

"Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

CMATO 3 – Low-Thrust LEO-to-GEO Trajectory Optimization

This *CMATO* application is a Fortran computer program named `leo2geo_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the continuous, single-maneuver, finite-burn low Earth orbit (LEO) to geosynchronous Earth orbit (GEO) orbit transfer optimization problem. The software attempts to maximize the final spacecraft mass. Since this simulation involves a single propulsive maneuver, this is equivalent to minimizing the propellant mass required for the orbital maneuver.

The important features of this scientific simulation are as follows:

- single, continuous thrust transfer trajectory
- constant propulsive thrust magnitude
- modified equinoctial equations of motion
- near-circular initial and final orbits
- two types of initial guess algorithms
- numerical verification of the optimal control solution

The *SOS* software suite is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *AMA_OC* software can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `leo2geo_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *SOS* software will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method. The `leo2geo_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `leo2geo_sos` software is "data-driven" by a user-created text file. Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly. In the following discussion the actual input file contents are in *courier* font and all explanations are in times font.

The following is a typical input file used by the `leo2geo_sos` computer program. This is a classic LEO-to-GEO example taken from the paper, "Minimum-Time Low-Thrust Rendezvous and Transfer Using Epoch Mean Longitude Formulation", Jean A. Kechichian, *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 3, May-June 1999. For this example, the initial true longitude is free. However, the SOS solution includes the effect of propellant mass depletion due to thrusting while the example in the technical paper assumes constant mass and therefore constant thrust acceleration. This example also includes the effect of the Earth's J_2 gravity term.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*********************************************
** earth orbit trajectory optimization with SOS
** single phase continous-thrust maneuver
** program leo2geo - leo-to-geo orbit transfer
** j. kechichian example - leo2geo_jk.in
*********************************************
```

The first three program inputs are the initial spacecraft mass, thrust magnitude and specific impulse.

```
initial spacecraft mass (kilograms)
1000.0

thrust magnitude (newtons)
98.0

specific impulse (seconds)
3300.0
```

The next six numerical inputs are the classical orbital elements of the initial orbit.

```
****************
* INITIAL ORBIT *
****************

semimajor axis (kilometers)
7000.0

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
```

```
0.0

true anomaly (degrees)
0.0
```

The following input determines if the software will constrain or free the initial true longitude.

```
constrain initial true longitude (1 = yes, 2 = no)
2
```

The following five inputs are the user-defined classical orbital elements of the final orbit.

```
***************
* FINAL ORBIT *
***************

semimajor axis (kilometers)
42000.0

orbital eccentricity (non-dimensional)
0.001

orbital inclination (degrees)
1.0

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
0.0
```

The next integer input defines the type of final orbit point constraints. Please see the "Problem setup" section for additional information about this program item.

```
*************************************
* type of final orbit point constraints *
-----------------------------------------
1 = modified equinoctial orbital elements (semimajor axis, ecc = 0, inc = 0)
2 = eci components of final state vector (hx, hy, hz, |r|, sin(gamma))
3 = all components of final classical orbital elements
-----------------------------------------------------
2
```

The next program input tells the software what type of gravity model to use during the simulation. Option 2 will include the oblateness gravity coefficient $\left(J_2\right)$ in the equations of motion.

```
************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = j2 gravity model
--------------------
2
```

The next integer input defines the type of initial guess used to estimate the transfer time. Please see the "initial guess" part of "Problem setup" section for additional information about this program item.

```
****************************************
* type of initial guess for transfer time *
----------------------------------------
1 = numerical integration (coplanar orbits)
2 = Edelbaum algorithm (non-coplanar orbits)
3 = user-defined
```

```
----------------
2
```

The following data item is the user's initial guess for the transfer time, in hours. This input corresponds to option 3 of the previous item.

```
*******************************************************
* user-defined initial guess for transfer time (hours) *
*******************************************************
4
```

The next integer input defines the type of initial guess to use for the simulation Please see the "Initial guess" part of "Problem setup" section for additional information about this program item.

```
*******************************
* initial guess/restart option *
*******************************
 1 = linear guess with tangential thrusting
 2 = numerical integration with Edelbaum steering
 3 = binary data file
---------------------
2
```

This next line contains the name of the binary data file used for the initial guess when the user selects the binary data file option.

```
name of initial guess/restart input data file
leo2geo_jk.rsbin
```

The next two program inputs determine if a binary file is updated or created and the name of this file.

```
****************************************
* restart and solution data file options *
****************************************

create/update binary restart data file (yes or no)
no

name of binary restart data file
leo2geo_jk.rsbin
```

This next input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by *SOS*. Options 2 and 3 allow the user to specify either the number of nodes or time step size of the data file.

```
*********************************************
* type of comma-delimited solution data file *
*********************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this next input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
50
```

The name of the solution data file is defined in this next line.

```
name of solution output file
leo2geo_jk.csv
```

The next series of program inputs are algorithm control options and parameters for the *SOS* software. The first input is an integer that specifies the type of collocation method to use during the solution process.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
 4 = Runge-Kutta 4-stage
------------------------
1
```

The next input is an integer that defines the number of grid points to use for the initial guess.

```
number of grid points
100
```

The next input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
500000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
1000
```

The level of output from the SOS NLP algorithm is controlled with the following integer input.

```
*************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the *SOS* optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the *SOS* differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
***************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the SOS user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

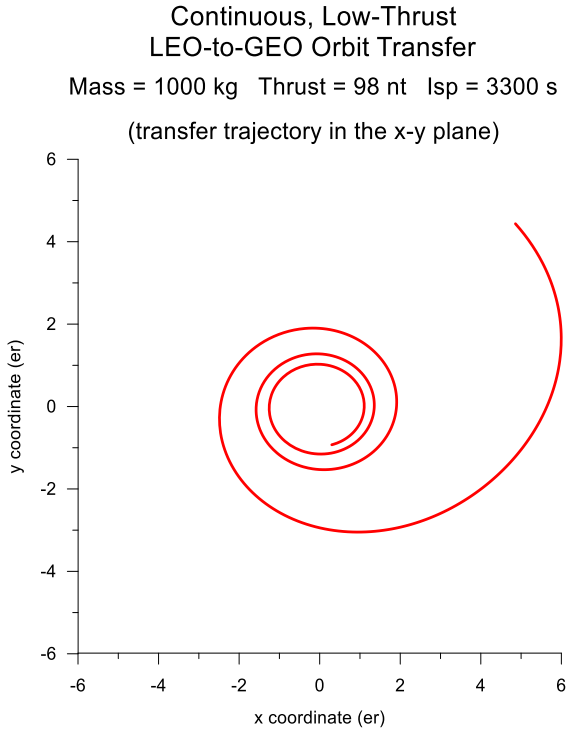
Optimal control solution

Here is the numerical information computed by the `leo2geo_sos` computer program for this example.

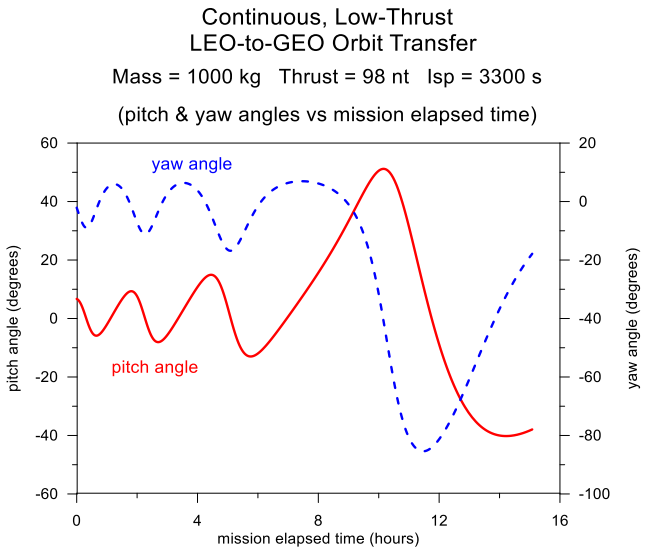
```
 ===================
 program leo2geo_ocs
 ===================

 transfer time algorithm
 -----------------------
 Edelbaum algorithm (non-coplanar orbits)

 initial guess type
 ------------------
 linear guess with tangential thrusting

 gravity model type
 ------------------
 j2 earth gravity model

 ------------------------
 beginning of finite burn
 ------------------------
```

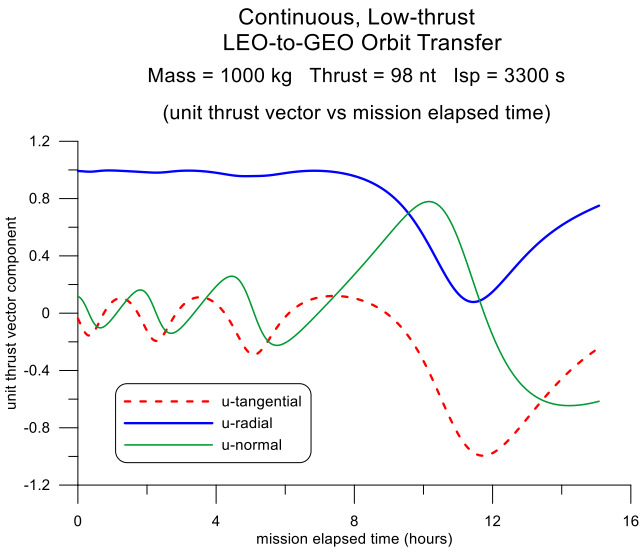
```
     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.700000000000D+04    0.615067684519D-16    0.285000000000D+02    0.000000000000D+00

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.206748308333D-14    0.285866456178D+03    0.285866456178D+03    0.971419368695D+02

     rx (km)                ry (km)               rz (km)               rmag (km)
 0.191377284137D+04    -.591734870255D+04    -.321285820480D+04    0.700000000000D+04

     vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 0.725856076669D+01    0.181305405204D+01    0.984408031306D+00    0.754605384101D+01

------------------
end of finite burn
------------------

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.420000000000D+05    0.999999999999D-03    0.100000000000D+01    0.360000000000D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.000000000000D+00    0.423922295262D+02    0.423922295262D+02    0.142768906774D+04

     rx (km)                ry (km)               rz (km)               rmag (km)
 0.309960418384D+05    0.282912582669D+05    0.493825749949D+03    0.419689619582D+05

     vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 -.207699129982D+01    0.227794898386D+01    0.397617474165D-01    0.308294103563D+01

transfer time          15.0823889600487       hours

spacecraft mass        835.576420850925       kilograms

propellant mass        164.423579149075       kilograms

delta-v                5813.28840908474       meters/second
```

This first plot is a view of the transfer trajectory from a north pole viewpoint looking down on the equatorial plane. The unit of each trajectory coordinate is Earth radii (ER).

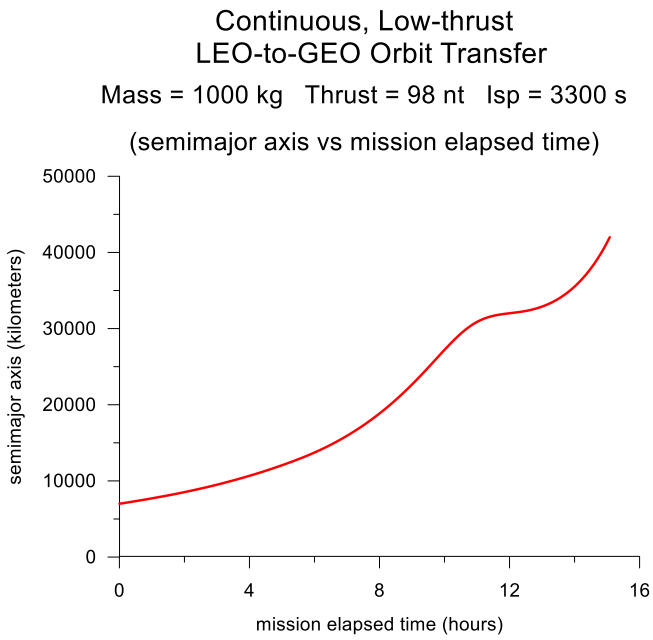


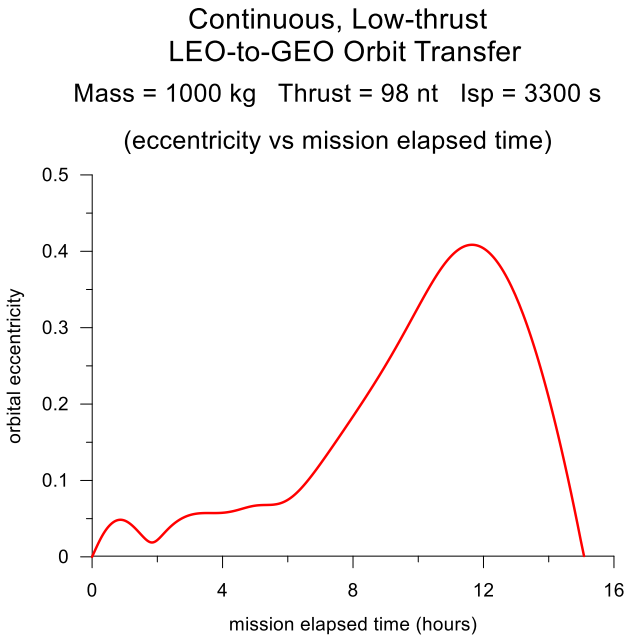
The following are typical optimal control and orbital element plots for this example. The first two plots illustrate the behavior of the pitch and yaw angles, and the components of the RTN unit thrust vector.



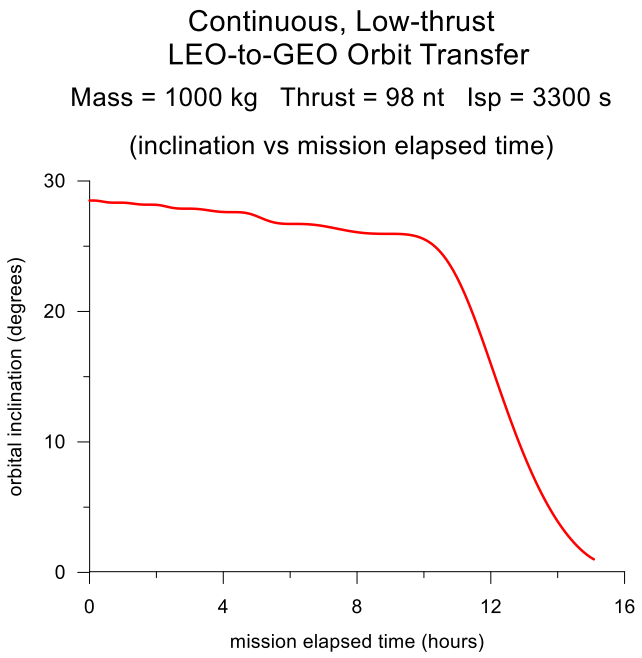


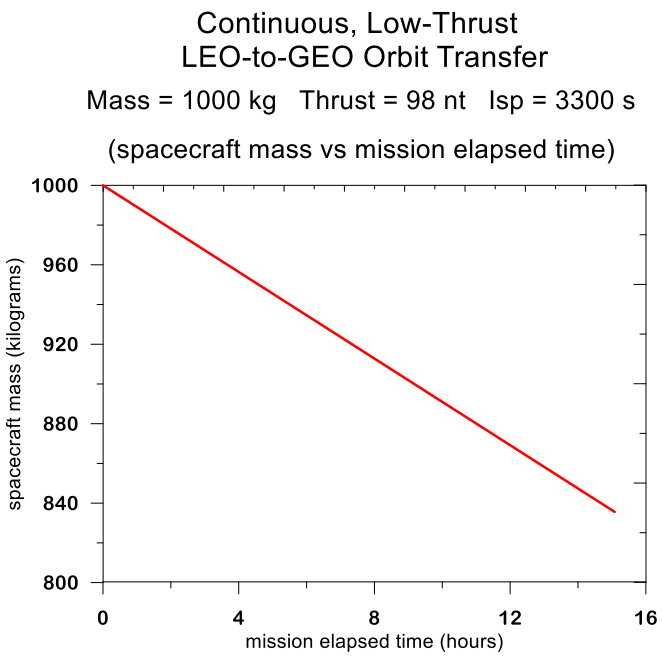
The next two plots illustrate the behavior of the semimajor axis and orbital eccentricity during the continuous thrust orbit transfer.





The final two plots show the behavior of the orbital inclination and the linear variation of the spacecraft mass.





The `leo2geo_sos` software will create a comma-separated-variable (csv) output file. This file contains the state vector, orbital elements, and steering angles during the transfer trajectory. The software will also display a summary of the final solution and a numerical verification of the optimal control solution.

Verification of the optimal control solution

The optimal control solution can be verified by numerically integrating the SOS-computed initial conditions and optimal control solution for the unit thrust vector to the final time determined by the *Sparse Optimization Suite*. This software uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is a typical display of the final solution and errors computed using the explicit numerical integration method. The errors are the differences between the spacecraft's final orbital elements and the user-defined final orbit.

```
--------------------
program leo2geo_sos
--------------------

transfer time algorithm
-----------------------
Edelbaum algorithm (non-coplanar orbits)

initial guess type
------------------
numerical integration with Edelbaum steering

gravity model type
------------------
j2 earth gravity model

-------------------------
beginning of finite burn
-------------------------

      sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.700000000000D+04    0.224492217480D-15    0.285000000000D+02    0.000000000000D+00

     raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
```

```
  0.360000000000D+03    0.105863244282D+03    0.105863244282D+03    0.971419368695D+02

      rx (km)               ry (km)               rz (km)              rmag (km)
  -.191339538138D+04    0.591744297501D+04    0.321290939057D+04    0.700000000000D+04

      vx (kps)              vy (kps)              vz (kps)             vmag (kps)
  -.725867640673D+01    -.181269645716D+01    -.984213873132D+00    0.754605384101D+01

------------------
end of finite burn
------------------

      sma (km)             eccentricity       inclination (deg)       argper (deg)
  0.420000000000D+05    0.100000000000D-02    0.100000000000D+01    0.222410091585D+03

     raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
  0.576918078643D-14    0.315511024401D-11    0.222410091585D+03    0.142768906774D+04

      rx (km)               ry (km)               rz (km)              rmag (km)
  -.309791255255D+05    -.282935265158D+05    -.493865342381D+03    0.419580000000D+05

      vx (kps)              vy (kps)              vz (kps)             vmag (kps)
  0.207977820170D+01    -.227649549166D+01    -.397363766156D-01    0.308374578579D+01

-------------------------------------------
integrated solution with sos optimal control
-------------------------------------------

transfer time          15.0845258071018       hours

spacecraft mass        835.553125600283       kilograms

propellant mass        164.446874399717       kilograms

delta-v                5814.19064868134       meters/second

final classical orbital element errors
--------------------------------------

semimajor axis         1.84608384006424       meters

eccentricity         -3.837974172564255E-008

inclination           7.129117037146725E-007  degrees

RAAN                 -9.054617670309940E-005  degrees
```

Creating an Initial guess

The solution to this classic orbit transfer problem requires a good initial guess for the transfer time along with a reasonable guess for the dynamic variables during the orbit transfer.

Transfer time initial guess

The `leo2geo_sos` software implements three options for specifying the transfer time initial guess. The first technique simply integrates the *coplanar* transfer orbit trajectory using tangential thrusting until the final orbit radius is reached. The second method uses Edelbaum's algorithm. The first method is best for coplanar orbit transfers and the second method should be used for non-coplanar orbit transfers. The third initial guess option is a user-specified transfer time.

For the first technique, the unit thrust vector for tangential steering during the numerical integration is simply $\mathbf{u}_T = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T$.

The Edelbaum algorithm is described in Chapter 14 of the book *Orbital Mechanics* by V. Chobotov and the technical paper, "The Reformulation of Edelbaum's Low-thrust Transfer Problem Using Optimal Control Theory" by J. A. Kechichian, AIAA-92-4576-CP. The original Edelbaum algorithm is described in "Propulsion Requirements for Controllable Satellites", ARS Journal, Aug. 1961, pp. 1079-1089. This algorithm is valid for total inclination changes Δi given by $0 < \Delta i < 114.6^\circ$ and assumes that the thrust acceleration magnitude and spacecraft mass are both constant during the orbit transfer.

The initial thrust vector yaw angle β_0 is given by the following expression

$$\tan \beta_0 = \frac{\sin\left(\frac{\pi}{2}\Delta i\right)}{\frac{V_0}{V_f} - \cos\left(\frac{\pi}{2}\Delta i\right)}$$

where the speed on the initial circular orbit is $V_0 = \sqrt{\mu / r_0}$ and the speed on the final circular orbit is $V_f = \sqrt{\mu / r_f}$. In these equations $r_0 = r_e + h_0$ is the geocentric radius of the initial orbit, $r_f = r_e + h_f$ is the geocentric radius of the final orbit, r_e is the radius of the Earth and μ is the gravitational constant of the Earth. The initial altitude is h_0 and the final altitude is h_f.

The total velocity change required for a low-thrust orbit transfer is given by

$$\Delta V = V_0 \cos \beta_0 - \frac{V_0 \sin \beta_0}{\tan\left(\frac{\pi}{2}\Delta i + \beta_0\right)}$$

The total transfer time is given by $t = \Delta V / f$ where f is the thrust acceleration. This is the transfer time used for the Edelbaum initial guess option in the `leo2geo_sos` software.

Dynamic variables initial guess

The initial guess for the dynamic variables is created by numerically integrating the equations of motion from the initial conditions to each individual grid point. For better algorithm performance, the initial grid points used by the `leo2geo_sos` software are equally distributed in *true longitude* instead of time. This technique will group more grid points around perigee of the elliptical transfer orbit.

The dynamical equations of motion with true longitude L as the independent variable can be determined from the following two expressions:

$$\dot{\mathbf{z}} = \frac{d\mathbf{z}}{dL}\frac{dL}{dt} = \mathbf{f}(\mathbf{z})$$

$$\frac{d\mathbf{z}}{dL} = \mathbf{f}(\mathbf{z})\left[\frac{dL}{dt}\right]^{-1} = \frac{1}{\dot{L}}\mathbf{f}(\mathbf{z})$$

where $\mathbf{z}$ is the vector of modified equinoctial orbital elements.

The true longitudes of the grid points are computed from

$$L_k = L_i + \frac{(k-1)}{(M-1)}\left(L_f - L_i\right) \qquad k = 1,\ldots,M$$

where L_i and L_f are the initial and final values of true longitude, and *M* is the number of user-specified grid points.

During the creation of dynamic variables by numerical integration, the components of the unit thrust vector are defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle β as follows

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\beta(t) \qquad u_{T_n} = \cos\theta\sin\beta(t)$$

where the pitch angle for tangential steering is simply $\theta = 0$ degrees.

According to the Edelbaum algorithm, the time evolution of the yaw angle is given by the following expression

$$\beta(t) = \tan^{-1}\left(\frac{V_0\sin\beta_0}{V_0\cos\beta_0 - f\,t}\right)$$

Problem setup

This section provides additional details about the `leo2geo_sos` software implementation.

Performance index – maximize final spacecraft mass

The objective function or performance index *J* for this simulation is the mass of the spacecraft when it reaches the final orbit and thrust has terminated. This is simply $J = m_f$.

The value of the `maxmin` indicator in *SOS* tells the software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the initial value provided by the user.

Path constraint – unit thrust vector scalar magnitude

At any point during the transfer trajectory, the scalar magnitude of the components of the unit thrust vector is constrained as follows:

$$\left|\mathbf{u}_T\right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

This formulation avoids problems that may occur when using thrust steering angles as control variables.

Initial true longitude

The software allows the use to either fix or free the initial true longitude. For an unconstrained initial true longitude, the true longitude bounds are $0 \le L \le 2\pi$. Otherwise, the initial true longitude is fixed to the value $L = (\Omega + \omega + \theta)_i$.

Mission constraint "matching" at the final orbit

The `leo2geo_sos` software implements three techniques for "targeting" the final mission orbit. The correct option to use depends on the characteristics of the final mission orbit.

For final orbits that are circular and equatorial, the set of final constraints are specified in terms of the final modified equinoctial elements or dynamic variables according to

$$p = p_f \qquad f_f = g_f = h_f = k_f = 0$$

The *f* subscript indicates values on the user-specified final orbit. This set of constraints or boundary conditions follows from the orbital element definitions

$$p = a\left(1-e^2\right) \qquad f = e\cos\left(\omega+\Omega\right) \qquad g = e\sin\left(\omega+\Omega\right)$$

$$h = \tan\left(i/2\right)\cos\Omega \qquad k = \tan\left(i/2\right)\sin\Omega$$

These boundary conditions are enforced using lower and upper bounds on the dynamic variables at the final time.

For final orbits that may be near-circular and/or inclined, the final mission constraints are enforced using *point functions*. This set of point functions is given by

$$h_x = h_{x_f} \qquad h_y = h_{y_f} \qquad h_z = h_{z_f} \qquad r = r_f \qquad \sin\gamma = \sin\gamma_f$$

where h_x, h_y, h_z are the inertial components of the angular momentum vector, *r* is the geocentric radius and γ is the flight path angle. As noted previously, the *f* subscript indicates mission values on the user-specified final orbit.

A third program option allows the user to constrain a classical orbital element set consisting of the semimajor axis, orbital eccentricity and inclination using point functions. This set of mission constraints is

$$a = a_f \qquad e = e_f \qquad i = i_f$$

As before the *f* subscript indicates values on the user-specified final orbit.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables during the orbital transfer.

$$0.05m_{sc_i} \le m_{sc} \le 1.05m_{sc_i} \qquad 1.1p_f \le p \le 0.9p_i$$

$$-1 \le f \le +1 \qquad -1 \le g \le +1 \qquad -1 \le h \le +1$$

$$-1 \le k \le +1 \qquad 0 \le L \le 200 \cdot 2\pi$$

where m_{sc_i} is the initial spacecraft mass, and p_i is the semi-parameter of the initial orbit and p_f is the semi-parameter of the final orbit. The upper bound on true longitude *L* allows for a maximum of 200 complete orbits during the transfer.

Finally, the components of the unit thrust vector are constrained as follows:

$$-1.1 \le u_r \le +1.1 \qquad -1.1 \le u_t \le +1.1 \qquad -1.1 \le u_n \le +1.1$$

Technical discussion

In this computer program, the orbital motion of the spacecraft is modeled in the modified equinoctial orbital elements coordinate system. Please consult *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for a definition of these orbital elements along with the equations of motion in this system.

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m}\hat{\mathbf{u}}_T$$

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_r} & u_{T_t} & u_{T_n} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system. *The components of this unit vector are the control variables.*

The propellant mass flow rate is determined from

$$\dot{m} = \frac{dm}{dt} = \frac{T}{g\, I_{sp}}$$

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g\, I_{sp}$ is also called the *exhaust velocity*. This differential equation and the modified equinoctial differential equations are included in the right-hand-side subroutine required by *SOS*. The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}\, t$ where m_{sc_i} is the initial mass of the spacecraft.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as follows

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

The pitch and yaw angles can be determined from the components of the unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

The pitch angle is positive above the "local horizontal" and the yaw angle is positive in the direction of the angular momentum vector of the transfer orbit.

Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `leo2geo_sos` software.

The simulation summary screen display contains the following information:

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (min) = orbital period in minutes

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

final mass = final spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The delta-v is determined using a cubic spline integration of the thrust acceleration data at each collocation node.

The comma-separated-variable disk file contains the following information.

time (hrs) = simulation time since ignition in hours

time (days) = simulation time since ignition in days

semimajor axis (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argument of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (min) = orbital period in minutes

mass (kg) = spacecraft mass in kilograms

T/W = thrust-to-weight ratio

yaw = thrust vector yaw angle in degrees

pitch = thrust vector pitch angle in degrees

perigee altitude = perigee altitude in kilometers

apogee altitude = apogee altitude in kilometers

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

semi-parameter = orbital semiparameter in kilometers

f equinoctial element = modified equinoctial orbital element

g equinoctial element = modified equinoctial orbital element

h equinoctial element = modified equinoctial orbital element

k equinoctial element = modified equinoctial orbital element

true longitude = true longitude in degrees

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

fpa (deg) = flight path angle in degrees

dvacc (mps) = accumulative delta-v in meters per second

"Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

"Direct Trajectory Optimization Using Nonlinear Programming and Collocation", C. R. Hargraves and S. W. Paris, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 10, No. 4, July-August, 1987, pp. 338-342.

"Survey of Numerical Methods for Trajectory Optimization", John T. Betts, AIAA *Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998.

CAMTO 4 – Finite-Burn, Earth Orbit Rendezvous Trajectory Optimization

This *CMATO* application is a Fortran computer program named `rendezvous_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the classic Earth orbit rendezvous trajectory optimization problem. The software models the trajectory as a mission consisting of one or more maneuvers separated by coasting periods. The propulsive phases are simulated as variable thrust, finite-burn propulsive maneuvers. This computer program attempts to maximize the spacecraft mass at the end of the propulsive maneuvers.

The important features of this scientific simulation are as follows.

- finite-burn orbital maneuvers
- variable attitude steering during all maneuvers
- user-defined throttle bounds
- modified equinoctial equations of motion with oblate Earth gravity model
- user-specified flyby or rendezvous final orbit constraints
- fixed or free final flyby or rendezvous time

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods.

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `rendezvous_sos` software consists of Fortran routines that perform the following tasks.

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method.

The `rendezvous_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `rendezvous_sos` software is "data-driven" by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in times font. This example attempts to optimize the maneuvers required to perform a rendezvous between a spacecraft in a circular low Earth orbit (LEO) and a second spacecraft in a typical medium altitude Earth orbit (MEO).

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
**********************************
** finite-burn earth-orbit rendezvous
** trajectory optimization
** program rendezvous_sos
** leo2meo_10k.in - April 10, 2012
**********************************
```

The first input is an integer that tells the simulation what type of trajectory to model.

```
trajectory type (1 = flyby, 2 = rendezvous)
2
```

The next three inputs define an initial guess, lower bound and upper bound for the total simulation duration in minutes. Identical values for the lower and upper bounds will create a fixed time mission.

```
initial guess for total simulation duration (minutes)
85.0

lower bound for total simulation duration (minutes)
88.0

upper bound for total simulation duration (minutes)
88.0
```

The next input is the initial mass of the entire spacecraft in kilograms.

```
initial spacecraft mass (kilograms)
8000.0
```

This next integer input defines the type of initial guess for the propulsive maneuver.

```
*******************************
type of propulsive initial guess
*******************************
1 = thrust duration
2 = delta-v magnitude
---------------------
2
```

The next four inputs define the thrust magnitude and the specific impulse of the upper stage or spacecraft propulsion system, and the user's initial guess for either the delta-v or thrust duration for the first maneuver.

```
-------------------------
first propulsive maneuver
-------------------------
thrust magnitude (newtons)
10000.0

specific impulse (seconds)
350.0

initial guess for delta-v (meters/second)
2925.0

initial guess for thrust duration (seconds)
170.0
```

The next six inputs define the classical orbital elements of the initial park orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
8000.0

orbital eccentricity (non-dimensional)
0.015

orbital inclination (degrees)
28.5

argument of perigee (degrees)
100.0

right ascension of the ascending node (degrees)
20.0

true anomaly (degrees)
30.0
```

The next six inputs define the classical orbital elements of the final mission orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
***************
* FINAL ORBIT *
***************

semimajor axis (kilometers)
10000.0

orbital eccentricity (non-dimensional)
0.05

orbital inclination (degrees)
40.0

argument of perigee (degrees)
200.0
```

```
right ascension of the ascending node (degrees)
55.0

true anomaly (degrees)
120.0
```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will use a J_2 gravity model in the spacecraft equations of motion.

```
************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate gravity model
------------------------
2
```

This next input defines the type of initial guess to use. Please see the Creating an initial guess section for information about how the first option is modeled. Option 2 requires either a binary restart file created from a previous run using either initial guess option 1 or an updated binary restart file. This feature is described in the next two sections.

```
************************
* initial guess options *
************************
 1 = numerical integration
 2 = binary data file
---------------------
1
```

If the user elects to use a binary data file (option 2 above) for the initial guess, the following text input specifies the name of the file to use.

```
name of binary initial guess data file
leo2meo_10k.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess option 1, the software will create a binary restart file. For initial guess option 2, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*****************************
* binary restart file option *
*****************************

create/update binary data file (yes or no)
no
```

This next input specifies the type of solution data file to create.

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = OC-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
--------------------------
1
```

For options 2 or 3, this input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
25
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
leo2heo_10k.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
---------------------------------
1
```

The next input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
10000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the *Sparse Optimization Suite* differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
****************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of user inputs allow the reading and writing of configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `rendezvous_sos` software. Please consult *Appendix J – Sparse Optimization Suite Configuration File* for additional information about these files.

```
************************************
* optimal control configuration options
************************************

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
leo2meo_10k_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
leo2meo_10k_config1.txt
```

Optimal control solution

The following is the optimal control solution for this example. The output includes the time and orbital characteristics at the beginning and end of each mission phase.

```
program rendezvous_sos
======================

input file ==> leo2meo_10k.in

rendezvous trajectory

oblate earth gravity model

---------------------------
beginning of maneuver phase
---------------------------

mission elapsed time     00:00:00.000

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.800000000000D+04    0.150000000000D-01    0.285000000000D+02    0.100000000000D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.200000000000D+02    0.300000000000D+02    0.130000000000D+03    0.118684693004D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.658713032027D+04    0.325905620287D+04    0.288604970418D+04    0.789563272225D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.381028378016D+01    -.564780744839D+01    -.217399960475D+01    0.715138208606D+01

---------------------
end of maneuver phase
---------------------

mission elapsed time     00:54:37.540

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.999778599805D+04    0.500760702724D-01    0.399921786418D+02    0.199752487246D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.550628877938D+02    0.467623343061D+02    0.246514821552D+03    0.165811819507D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.335362513193D+04    -.702990130399D+04    -.568340909816D+04    0.964196312202D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.515138258024D+01    0.360501455968D+01    -.181069217135D+01    0.654304811253D+01

spacecraft mass          3339.96227073119      kilograms

propellant mass          4660.03772926881      kilograms

phase duration           3277.53953053337      seconds
                         54.6256588422229      minutes

delta-v                  2998.07572342380      meters/second
```

The following program output is the spacecraft mass, the propellant mass consumed, the actual thrust duration for all the maneuvers, and the accumulated delta-v for the mission.

```
spacecraft mass          3339.96227073119      kilograms
```

```
propellant mass          4660.03772926881      kilograms

phase duration           3277.53953053337      seconds
                         54.6256588422229      minutes

delta-v                  2998.07572342380      meters/second
```

This section of the numeric results summarizes the time and orbital conditions at the beginning and end of the transfer orbit coast.

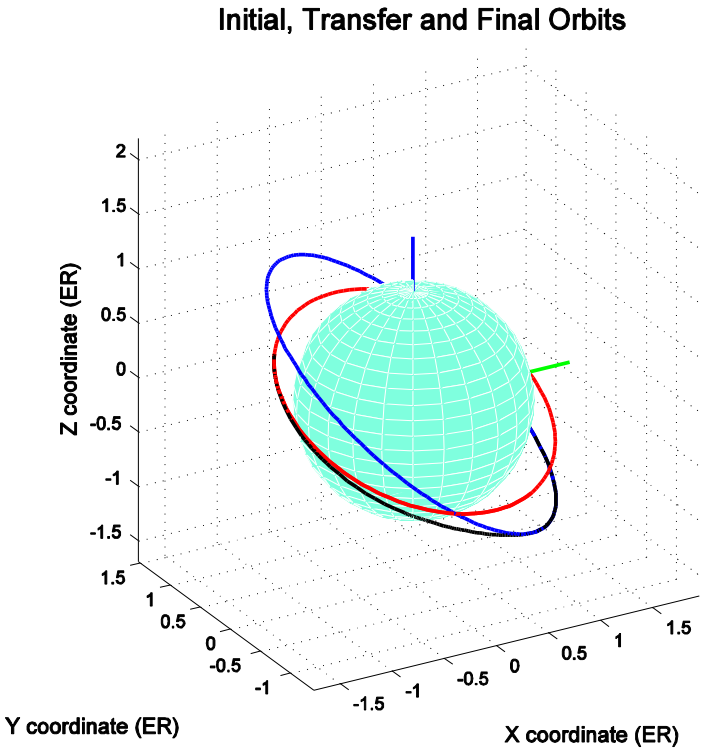
```
------------------------
beginning of coast phase
------------------------

mission elapsed time     00:54:37.540

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.999778599805D+04    0.500760702724D-01    0.399921786418D+02    0.199752487246D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.550628877938D+02    0.467623343061D+02    0.246514821552D+03    0.165811819507D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.335362513193D+04    -.702990130399D+04    -.568340909816D+04    0.964196312202D+04
     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.515138258024D+01    0.360501455968D+01    -.181069217135D+01    0.654304811253D+01

------------------
end of coast phase
------------------

mission elapsed time     01:27:60.000

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.100000000000D+05    0.500000000000D-01    0.400000000000D+02    0.200000000000D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.550000000000D+02    0.120000000000D+03    0.320000000000D+03    0.165866900904D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.862186499607D+04    0.353038894193D+04    -.422710739886D+04    0.102307692308D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.459681283666D+00    0.541422590670D+01    0.292176275871D+01    0.616942839083D+01

coast duration           2002.46046946663      seconds
                         33.3743411577771      minutes
```

After the simulation is complete, the software will display a simulation summary like the following

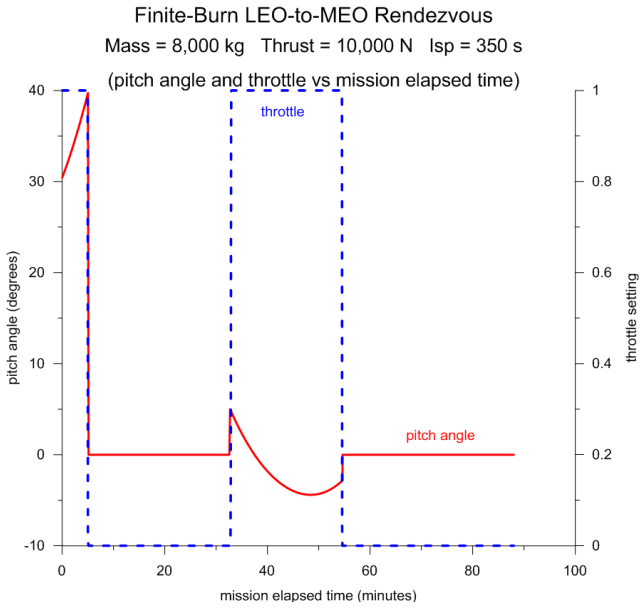
```
SIMULATION SUMMARY
==================

initial mass             8000.00000000000      kilograms

total propellant mass    4660.03772926881      kilograms

final spacecraft mass    3339.96227073119      kilograms

total delta-v            2998.07572342380      meters/second

total sim duration       5280.00000000000      seconds
                         88.0000000000000      minutes
```

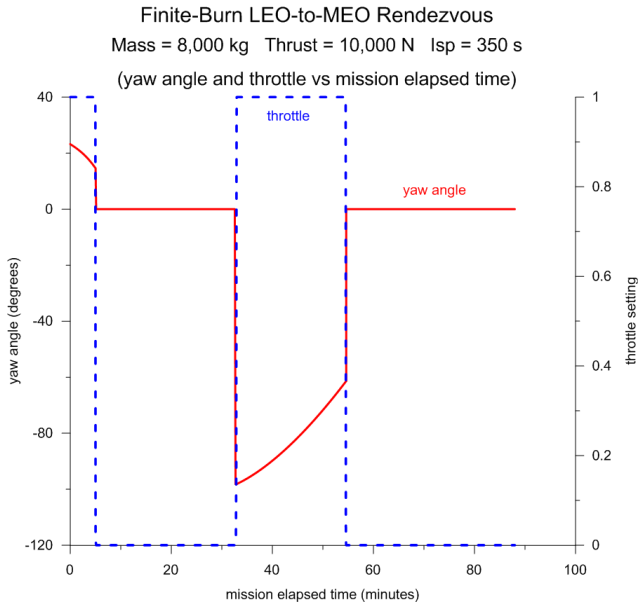
The `rendezvous_sos` computer program will also create an output file named `orbits.csv`. This file contains the Earth-centered inertial (ECI) position and velocity vectors of the park orbit and final mission orbit.

The following is the graphics display for this example. The initial orbit trace is red, the final orbit is blue and the transfer orbit is black. The dimensions are Earth radii (ER) and the plot is labeled with an ECI coordinate system where green is the x-axis, red is the y-axis and blue is the z-axis.

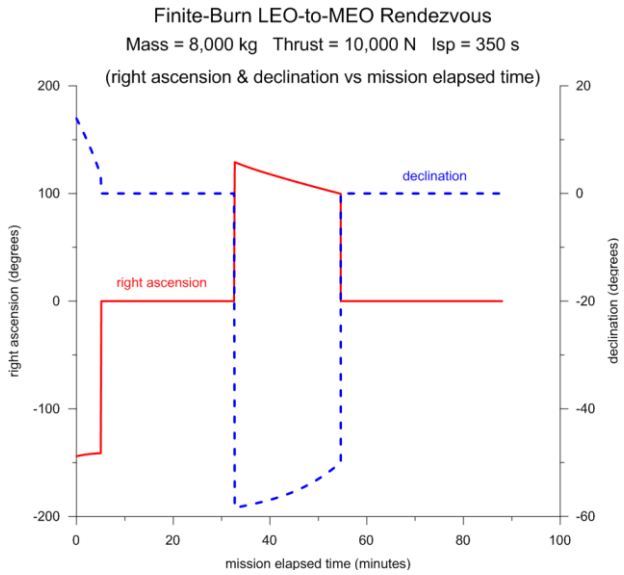


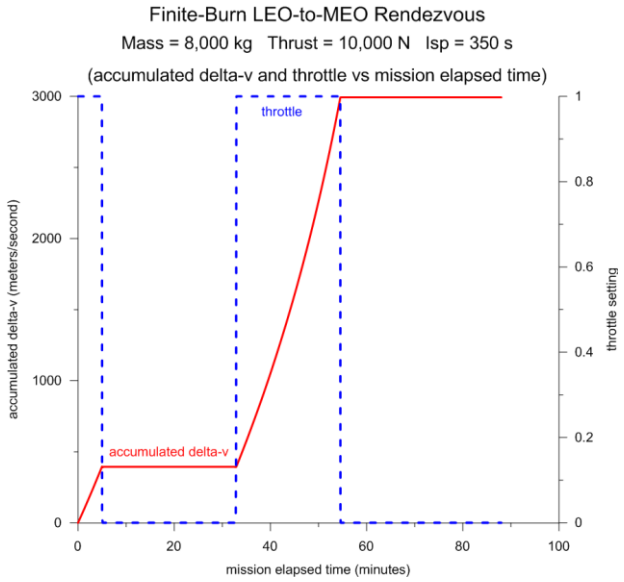
The following two plots illustrate the evolution of the pitch and yaw steering angles and the throttle setting during the two propulsive maneuvers determined by the software.



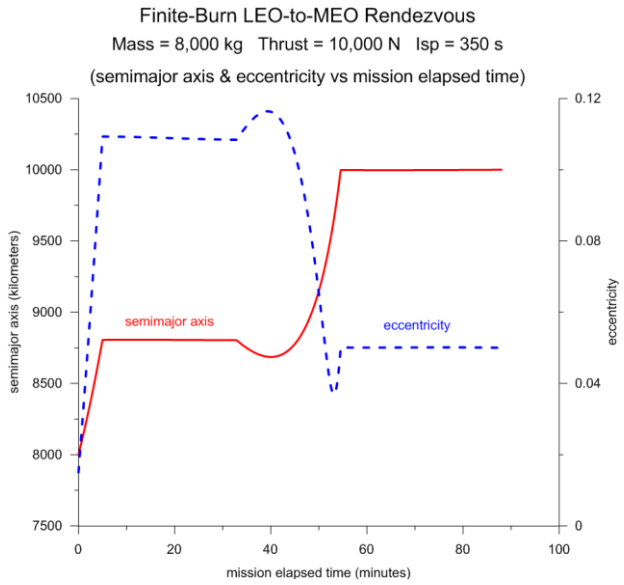


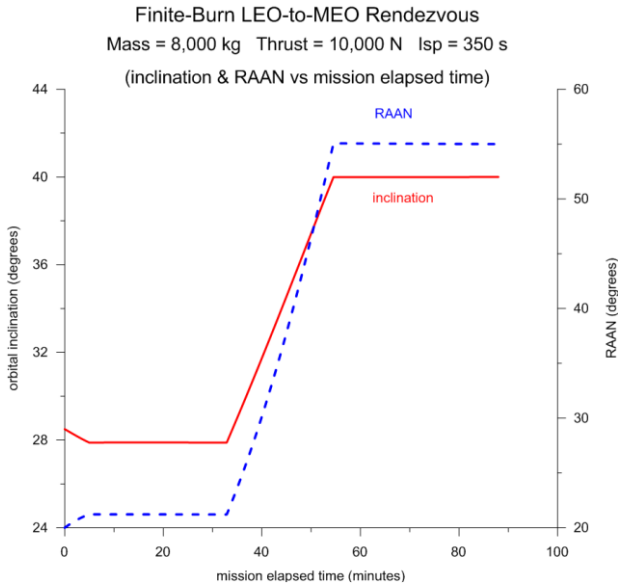
The next two plots illustrate the behavior of the inertial right ascension and declination, and the accumulated delta-v and throttle setting during the simulation.





The final pair of plots illustrate the behavior of the semimajor axis, eccentricity, inclination, and the right ascension of the ascending node (RAAN) during the simulation.





Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* can be verified by numerically integrating the orbital equations of motion with the OC-computed optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the optimal unit thrust vector solution. This part of the `rendezvous_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is a display of the final solution computed using this *explicit* numerical integration method.

```
verification of optimal control solution
========================================

final mass                3339.96216840498      kilograms

propellant mass           4660.03783159502      kilograms

delta-v                   2998.07650071517      meters/second

final mission orbit
-------------------

mission elapsed time      01:27:60.000

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.100000037673D+05    0.500000524679D-01    0.400000018124D+02    0.199999582004D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.549999947879D+02    0.120000292543D+03    0.319999874547D+03    0.165866994633D+03

     rx (km)                 ry (km)              rz (km)               rmag (km)
 0.862187310660D+04    0.353037110635D+04    -.422712123100D+04    0.102307756263D+05

     vx (kps)                vy (kps)             vz (kps)              vmag (kps)
 -.459669452953D+00    0.541422750064D+01    0.292175590011D+01    0.616942566002D+01
```

Creating an initial guess

The software allows the user to input either a delta-v or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. An estimate of the thrust duration can be determined from the following expression

$$t_d = \frac{I_{sp} m_p g}{F} = \frac{m_p V_{ex}}{F}$$

The propellant mass required for a given ΔV is a function of the initial (or final) mass of the spacecraft and the exhaust velocity as follows

$$m_p = m_i \left(1 - e^{\frac{-\Delta V}{V_{ex}}} \right) = m_f \left(e^{\frac{\Delta V}{V_{ex}}} - 1 \right)$$

In these equations

$m_i =$ initial mass

$m_f =$ final mass

$m_p =$ propellant mass

$V_{ex} =$ exhaust velocity $= g I_{sp}$

$I_{sp} =$ specific impulse

$\Delta V =$ impulsive velocity increment

$F =$ thrust

$g =$ acceleration of gravity

The software uses a tangential thrusting steering method to generate an initial guess for the optimal trajectory. For tangential thrusting, the unit thrust vector in the modified equinoctial frame is simply $\mathbf{u}_T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$.

The dynamic variables and control variables at each grid point are determined by the *Sparse Optimization Suite* by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2`. These program options create an initial guess from the numerical integration of the equations programmed in the `oderhs` subroutine. The number and location of the initial collocation nodes are determined from the variable step-size numerical integration. The `INIT(2) = 2` option tells the program to use a Dormand-Prince numerical integration method. Please note this algorithm creates a *coplanar* initial guess.

If the software cannot find a feasible solution, try increasing the guess for the thrust duration or the value for the magnitude of the delta-v.

Binary restart data files can also be used to initialize a `rendezvous_sos` simulation. A typical scenario is

- Create a binary restart file from a converged and optimized simulation
- Modify the original input file with slightly different spacecraft characteristics, propulsive parameters or perhaps final mission targets and/or constraints
- Use the previously created binary restart file as the initial guess for the new simulation

This technique works well provided the two simulations are not dramatically different. Sometimes it may be necessary to make successive small changes in the mission definition and run multiples simulations to eventually reach the final desired solution.

Problem setup

This section provides additional details about the software implementation. It explains such things as point and path constraints, the performance index and the numerical technique used to create an initial guess for the software.

Point functions – initial orbit constraints

For this two-point boundary value problem (TPBVP), both lower and upper bounds for all modified equinoctial elements are set equal to the user-defined initial modified equinoctial orbital elements as follows

$$p_L = p_U = p_i \qquad f_L = f_U = f_i$$

$$g_L = g_U = g_i \qquad h_L = h_U = h_i$$

$$k_L = k_U = k_i \qquad L_L = L_U = L_i$$

In *Sparse Optimization Suite* terminology, these constraints or boundary conditions are called *point functions*.

Performance index – maximize final spacecraft mass

The objective function or performance index *J* for this simulation is the mass of the spacecraft at the end of the mission. This is simply $J = m_f$.

The value of the `maxmin` indicator tells the software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the user-defined initial value.

Path constraint – unit thrust vector scalar magnitude

For variable attitude steering, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as follows:

$$\left|\mathbf{u}_T\right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

Point functions – final mission orbit constraints

The final mission constraints enforced by the software are determined by the trajectory type. For the flyby trajectory option, the final orbit position vector is constrained to the values corresponding to the user-defined final orbit. The rendezvous trajectory option adds the final three components of the inertial velocity vector to this constraint set.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the orbital transfer.

$$0.05m_{sc_i} \le m_{sc} \le 1.05m_{sc_i} \qquad 100\,p_f \le p \le 0.8\,p_i$$

$$-1 \le f \le +1 \qquad -1 \le g \le +1$$

$$-1 \le h \le +1 \qquad -1 \le k \le +1$$

where m_{sc_i} is the initial spacecraft mass.

For variable attitude steering, the three components of the unit thrust vector are constrained as follows:

$$-1.1 \le u_r \le +1.1$$

$$-1.1 \le u_t \le +1.1$$

$$-1.1 \le u_n \le +1.1$$

Technical Discussion

In this computer program, the orbital motion of the spacecraft is modeled in the modified equinoctial orbital elements coordinate system. Please consult *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for a definition of these orbital elements along with the equations of motion in this system.

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)}\hat{\mathbf{u}}_T$$

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_r} & u_{T_t} & u_{T_n} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system. *The components of this unit vector are the control variables.*

The propellant mass flow rate is determined from

$$\dot{m} = \frac{dm}{dt} = \frac{T}{g\, I_{sp}}$$

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g\, I_{sp}$ is also called the *exhaust velocity*. The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}\, t$ where m_{sc_i} is the initial mass of the spacecraft and $\dot{m}$ is the propellant flow rate.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

Finally, the pitch and yaw angles can be determined from the components of the unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the "local horizontal" and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four-quadrant operation. Please consult *Appendix G – Aerospace Trajectory Coordinates and Time Systems* for additional information.

The `rendezvous_sos` software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems. The following section summarizes the inertial-to/from-modified equinoctial coordinate transformations and the calculation of the inertial unit thrust vector in terms of right ascension and declination angles.

In this computer program, the components of the inertial unit thrust vector are defined in terms of the right ascension α and the declination angle δ as follows

$$u_{T_{ECI_x}} = \cos\alpha\cos\delta \qquad u_{T_{ECI_y}} = \sin\alpha\cos\delta \qquad u_{T_{ECI_z}} = \sin\delta$$

Finally, the right ascension and declination angles can be determined from the components of the ECI unit thrust vector according to $\alpha = \tan^{-1}\left(u_{T_{ECI_y}}, u_{T_{ECI_x}}\right)$ and $\delta = \sin^{-1}\left(u_{T_{ECI_z}}\right)$ where the calculation for right ascension is a four-quadrant inverse tangent operation.

Example flyby trajectory analysis

This section summarizes a flyby mission from LEO to MEO. This example starts and ends at the same orbits as the previous example. However, the propulsive thrust is 5000 Newtons and the total simulation time is fixed to 100 minutes. Furthermore, since this is a flyby trajectory, the orbit transfer only matches the three components of the position vector of the final mission orbit at the final time.

Here is the initial part of the simulation definition input file for this example.

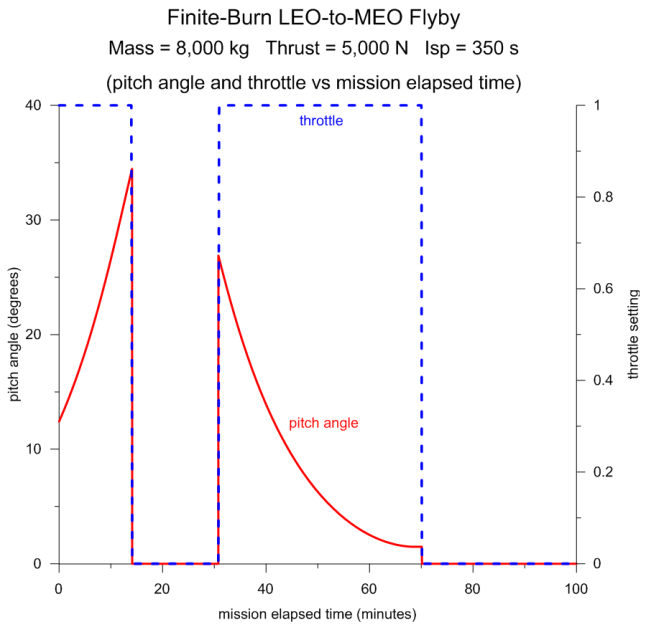
```
************************************
** finite-burn earth-orbit rendezvous
** trajectory optimization
** program rendezvous_sos
** leo2meo_flyby.in - April 10, 2012
************************************

trajectory type (1 = flyby, 2 = rendezvous)
1

initial guess for total simulation duration (minutes)
95.0

lower bound for total simulation duration (minutes)
100.0

upper bound for total simulation duration (minutes)
100.0

initial spacecraft mass (kilograms)
8000.0

****************************************
type of propulsive maneuver initial guess
****************************************
1 = thrust duration
2 = delta-v magnitude
---------------------
2

-------------------
propulsive maneuver
-------------------

thrust magnitude (newtons)
5000.0

specific impulse (seconds)
350.0

initial guess for delta-v (meters/second)
2925.0

initial guess for thrust duration (seconds)
170.0

lower bound for throttle setting
0.0

upper bound for throttle setting
1.0
```

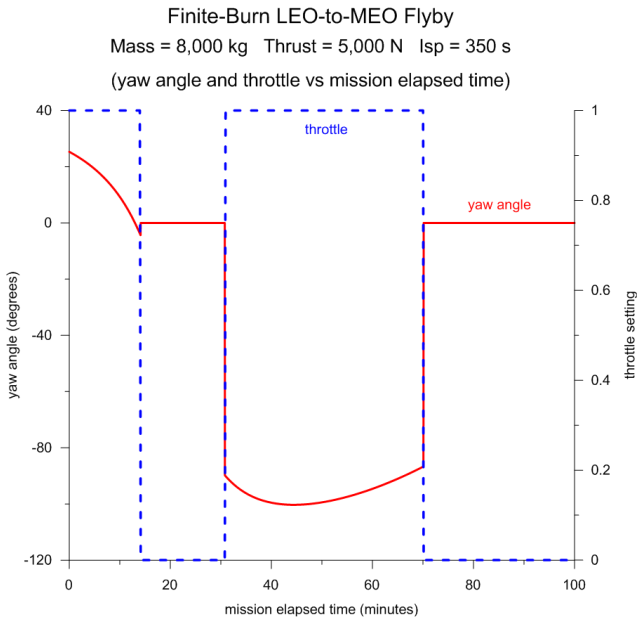
The program output for this example is next.

```
program rendezvous_sos
======================

input file ==> leo2meo_flyby.in

flyby trajectory

oblate earth gravity model

---------------------------
beginning of maneuver phase
---------------------------

mission elapsed time     00:00:00.000

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.800000000000D+04    0.150000000000D-01    0.285000000000D+02    0.100000000000D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.200000000000D+02    0.300000000000D+02    0.130000000000D+03    0.118684693004D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.658713032027D+04    0.325905620287D+04    0.288604970418D+04    0.789563272225D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.381028378016D+01    -.564780744839D+01    -.217399960475D+01    0.715138208606D+01

---------------------
end of maneuver phase
---------------------

mission elapsed time     01:10:11.736

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.898711286539D+04    0.235466089640D+00    0.350916114708D+02    0.903365417492D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.625248414158D+02    0.172979511791D+03    0.263316053540D+03    0.141315573076D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.739228594994D+04    -.529738113942D+04    -.632512076131D+04    0.110776934037D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.298144849526D+01    0.430206674637D+01    -.463960881703D+00    0.525471912855D+01

spacecraft mass          3343.59238906710      kilograms

propellant mass          4656.40761093290      kilograms

phase duration           4211.73581241822      seconds
                         70.1955968736369      minutes

delta-v                  2994.34791421085      meters/second

------------------------
beginning of coast phase
------------------------

mission elapsed time     01:10:11.736

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.898711286539D+04    0.235466089640D+00    0.350916114708D+02    0.903365417492D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.625248414158D+02    0.172979511791D+03    0.263316053540D+03    0.141315573076D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.739228594994D+04    -.529738113942D+04    -.632512076131D+04    0.110776934037D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.298144849526D+01    0.430206674637D+01    -.463960881703D+00    0.525471912855D+01
```

```
------------------
end of coast phase
------------------

mission elapsed time     01:40:00.000

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.898867629134D+04    0.235360368321D+00    0.351008512173D+02    0.903369639672D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.624745498836D+02    0.223728548707D+03    0.314065512674D+03    0.141352450221D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.862186499607D+04    0.353038894193D+04    -.422710739886D+04    0.102307692308D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.192043174636D+01    0.473402281402D+01    0.273461673498D+01    0.579458012140D+01

coast duration           1788.26418758178      seconds
                         29.8044031263631      minutes

SIMULATION SUMMARY
==================

initial mass             8000.00000000000      kilograms

total propellant mass    4656.40761093290      kilograms

final spacecraft mass    3343.59238906710      kilograms

total delta-v            2994.34791421085      meters/second

total sim duration       6000.00000000000      seconds
                         100.000000000000      minutes

verification of optimal control solution
========================================

final mass               3343.59248030948      kilograms

propellant mass          4656.40751969052      kilograms

delta-v                  2994.34782001675      meters/second

final mission orbit
-------------------

mission elapsed time     01:40:00.000

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.898867644662D+04    0.235360359825D+00    0.351008505110D+02    0.903369671277D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.624745489581D+02    0.223728544739D+03    0.314065511867D+03    0.141352453884D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.862186536286D+04    0.353038874660D+04    -.422710749938D+04    0.102307695140D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.192043150637D+01    0.473402286160D+01    0.273461656678D+01    0.579458000135D+01
```

Here are plots of the behavior of the control variables and throttle setting for this example.





Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `rendezvous_sos` software.

The simulation summary screen display contains the following information:

mission elapsed time = simulation time since beginning of the mission (hh:mm:ss.sss)

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (min) = orbital period in minutes

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

spacecraft mass = current spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

phase duration = current phase duration in seconds and minutes

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The accumulated delta-v is determined using a cubic spline integration of the thrust acceleration data at each collocation node.

The comma-separated-variable disk file is created by the `odeprt` subroutine and contains the following information.

time (sec) = simulation time since ignition in seconds

semimajor axis (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argument of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (min) = orbital period in minutes

mass = spacecraft mass in kilograms

thracc = thrust acceleration in meters/second**2

yaw = thrust vector yaw angle in degrees

pitch = thrust vector pitch angle in degrees

perigee altitude = perigee altitude in kilometers

apogee altitude = apogee altitude in kilometers

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

semi-parameter = orbital semiparameter in kilometers

f equinoctial element = modified equinoctial orbital element

g equinoctial element = modified equinoctial orbital element

h equinoctial element = modified equinoctial orbital element

k equinoctial element = modified equinoctial orbital element

true longitude = true longitude in degrees

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

fpa (deg) = flight path angle in degrees

deltav (mps) = accumulative delta-v in meters per second

The `orbits.csv` file contains the following information.

time (seconds) = simulation time since ignition in seconds

rp1-x (er) = x-component of the initial orbit position vector in earth radii

rp1-y (er) = y-component of the initial orbit position vector in earth radii

rp1-z (er) = z-component of the initial orbit position vector in earth radii

rp2-x (er) = x-component of the final orbit position vector in earth radii

rp2-y (er) = y-component of the final orbit position vector in earth radii

rp2-z (er) = z-component of the final orbit position vector in earth radii

Spacecraft Mission Design, Charles D. Brown, AIAA Education Series, 1992.

An Introduction to the Mathematics and Methods of Astrodynamics, Richard H. Battin, AIAA Education Series, 1987.

"Survey of Numerical Methods for Trajectory Optimization", John T. Betts, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998, pp. 193-207.

"Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

"Optimal Maneuvers of Orbital Transfer Vehicles", John M. Hanson, Ph.D. Thesis, University of Michigan, Department of Aerospace Engineering, 1983.

CMATO 5 – Atmospheric Reentry Trajectory Optimization

This *CAMTO* application is a Fortran computer program named `reentry_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the atmospheric reentry trajectory optimization problem. The trajectory is modeled as a single phase with user-defined initial and final boundary conditions. This computer attempts to maximize either the crossrange or downrange distance. The type of optimization is selected by the user.

The important features of this scientific simulation are as follows:

- angle-of-attack and bank angle control variables
- 3-DOF flight path equations of motion relative to a spherical, rotating Earth
- U.S. Standard 1976 atmosphere model and second-order zonal gravity model
- user-defined vehicle aerodynamic and mass properties
- user-defined mission constraints such as load factor, heat rate, etc.
- numerical verification of the optimal control solution

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `reentry_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method.

The `reentry_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `reentry_sos` software is "data-driven" by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in times font.

This data file defines a Space Shuttle maximum cross range reentry trajectory simulation with a maximum heat rate path constraint. For this example, the Terminal Area Energy Management (TAEM) conditions are fixed. Furthermore, all flight conditions except flight path angle at the Entry Interface (EI) are also fixed. The flight path angle at the Entry Interface is bounded according to $-10^\circ \le \gamma \le -0.5^\circ$ and the software selects the best flight path angle for maximum cross range subject to the heat rate path constraint. The bank angle is constrained according to $-89^\circ \le \psi \le +1^\circ$ which forces the vehicle to bank left while maximizing the cross-range.

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
****************************************
atmospheric reentry trajectory optimization
simulation definition data file
April 23, 2012
STS maximum crossrange w/ qdot constraint
****************************************
```

The first program input is an integer that defines the type of mission to simulate.

```
******************
* simulation type *
******************
 1 = maximize downrange distance
 2 = maximize crossrange distance
--------------------------------
2
```

This section allows the user to define the vehicle weight, aerodynamic reference area and the nose radius used in the aero-heating calculations.

```
************************************************
* vehicle weight and aerodynamics characteristics *
************************************************

vehicle weight (pounds)
203000.0

aerodynamic reference area (square feet)
2690.0

nose radius (feet)
1.0
```

The following series of data items are reserved for user-defined initial conditions. To fix one or more initial conditions, the user should input identical lower and upper bounds. To free or un-constrain one or more initial states set the lower and/or upper bounds to `1.0d99`. Please note the units and valid data range for each item.

```
*************************************
* initial flight conditions and bounds *
*************************************
NOTE 1: set upper and lower bounds to
the initial value to constrain or
"fix" a flight condition.
NOTE 2: set bound to 1.0d99 to ignore
-------------------------------------

initial time (seconds)
0.0

upper bound for initial time (seconds)
0.0

lower bound for initial time (seconds)
0.0

initial altitude (feet)
400000.0

upper bound for initial altitude (feet)
400000.0

lower bound for initial altitude (feet)
400000.0

initial velocity (feet/second)
25600.0

upper bound for initial velocity (feet/second)
25600.0

lower bound for initial velocity (feet/second)
25600.0

initial flight path angle (-90 <= fpa <= +90; degrees)
-1.0

upper bound for initial flight path angle (-90 <= fpa <= +90; degrees)
-0.5

lower bound for initial flight path angle (-90 <= fpa <= +90; degrees)
-10.0

initial flight azimuth (0 <= azimuth <= 360; degrees)
90.0

upper bound for initial flight azimuth (0 <= azimuth <= 360; degrees)
90.0

lower bound for initial flight azimuth (0 <= azimuth <= 360; degrees)
90.0

initial declination (-90 <= declination <= +90; degrees)
0.0

upper bound for initial declination (-90 <= declination <= +90; degrees)
0.0
```

```
lower bound for initial declination (-90 <= declination <= +90; degrees)
0.0

initial east longitude (0 <= longitude <= 360; degrees)
0.0

upper bound for initial east longitude (0 <= longitude <= 360; degrees)
0.0

lower bound for initial east longitude (0 <= longitude <= 360; degrees)
0.0
```

The following series of data items allow the user to define the final flight conditions. To fix one or more conditions, the user should input identical lower and upper bounds. To free or un-constrain one or more final states set the lower and/or upper bounds to `1.0d99`. Again, please note the units and valid data range for each item.

```
************************************
* final flight conditions and bounds *
************************************
NOTE 1: set upper and lower bounds
to the final value to constrain or
"fix" a flight condition.
NOTE 2: set bound to 1.0d99 to ignore
------------------------------------

final time (seconds)
1800.0

upper bound for final time (seconds)
1.0d99

lower bound for final time (seconds)
1.0d99

final altitude (feet)
80000.0

upper bound for final altitude (feet)
80000.0

lower bound for final altitude (feet)
80000.0

final velocity (feet/second)
2500.0

upper bound for final velocity (feet/second)
2500.0

lower bound for final velocity (feet/second)
2500.0

final flight path angle (-90 <= fpa <= +90; degrees)
-5.0

upper bound for final flight path angle (-90 <= fpa <= +90; degrees)
-5.0

lower bound for final flight path angle (-90 <= fpa <= +90; degrees)
-5.0

final flight azimuth (0 <= azimuth <= 360; degrees)
90.0
```

```
upper bound for final flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

lower bound for final flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

final declination (-90 <= declination <= +90; degrees)
0.0

upper bound for final declination (-90 <= declination <= +90; degrees)
1.0d99

lower bound for final declination (-90 <= declination <= +90; degrees)
1.0d99

final east longitude (0 <= longitude <= 360; degrees)
90.0

upper bound for final east longitude (0 <= longitude <= 360; degrees)
1.0d99

lower bound for final east longitude (0 <= longitude <= 360; degrees)
1.0d99
```

The next series of data inputs define lower and upper bounds on the state variables during the mission phase. To free or un-constrain one or more states set the lower and/or upper bounds to `1.0d99`.

```
****************************************************************
* upper and lower bounds on the flight conditions during phase *
****************************************************************
NOTE: set bound to 1.0d99 to ignore
-----------------------------------

upper bound for altitude (feet)
450000.0d0

lower bound for altitude (feet)
1.0

upper bound for velocity (feet/second)
26000.0d0

lower bound for velocity (feet/second)
1.0

upper bound for flight path angle (-90 <= fpa <= +90; degrees)
+1.0

lower bound for flight path angle (-90 <= fpa <= +90; degrees)
-89.0

upper bound for flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

lower bound for flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

upper bound for declination (-90 <= declination <= +90; degrees)
+89.0

lower bound for declination (-90 <= declination <= +90; degrees)
-89.0

upper bound for east longitude (0 <= longitude <= 360; degrees)
1.0d99
```

```
lower bound for east longitude (0 <= longitude <= 360; degrees)
1.0d99
```

This section of the input file defines the user's initial guess and bounds for the control variables. To free or un-constrain one or more control variables set the lower and/or upper bounds to `1.0d99`.

```
************************************
* initial flight controls and bounds *
************************************
NOTE 1: set upper and lower bounds
to the initial value to constrain
or "fix" a flight control.
NOTE 2: set bound to 1.0d99 to ignore
------------------------------------

initial angle-of-attack (degrees)
25.0

upper bound for initial angle-of-attack (degrees)
+90.0

lower bound for initial angle-of-attack (degrees)
-90.0d0

initial bank angle (degrees)
-10.0d0

upper bound for initial bank angle (degrees)
+1.0d0

lower bound for initial bank angle (degrees)
-89.0d0
```

This section of the input file defines the final guesses and bounds for the control variables. To free one or more control variables set the lower and/or upper bounds to `1.0d99`.

```
**********************************
* final flight controls and bounds *
**********************************
NOTE 1: set upper and lower bounds
to the final value to constrain
or "fix" a flight control.
NOTE 2: set bound to 1.0d99 to ignore
------------------------------------

final angle-of-attack (degrees)
10.0

upper bound for final angle-of-attack (degrees)
+90.0

lower bound for final angle-of-attack (degrees)
-90.0

final bank angle (degrees)
0.0d0

upper bound for final bank angle (degrees)
+1.0d0

lower bound for final bank angle (degrees)
-89.0d0
```

This section of the input file defines the bounds for the control variables during the mission phase. To free or un-constrain one or more control variables, set the lower and/or upper bounds to `1.0d99`.

```
************************************************************
* upper and lower bounds on the flight controls during phase *
************************************************************
NOTE: set bound to 1.0d99 to ignore
----------------------------------

upper bound for angle-of-attack (degrees)
+90.0d0

lower bound for angle-of-attack (degrees)
-90.0d0

upper bound for bank angle (degrees)
+1.0d0

lower bound for bank angle (degrees)
-89.0d0
```

This next section allows the user to define and enforce one or more path constraints during the reentry phase. These constraints are enforced at all points along the reentry trajectory. The user should be careful not to enforce constraints that are inconsistent with either the initial and/or final boundary conditions

```
**********************
* flight constraints *
**********************

enforce a heating rate constraint (yes or no)
yes

heating rate upper bound constraint value (BTU/foot**2-second)
100.0d0

enforce a dynamic pressure constraint (yes or no)
no

dynamic pressure upper bound constraint value (pounds per square foot)
250.0d0

enforce a load factor constraint (yes or no)
no

load factor upper bound constraint value (non-dimensional)
1.1
```

This next input defines the type of initial guess to use. Please see the technical discussion section for information about how the first option is modeled. Option 2 requires either a binary restart file created from a previous run using either initial guess option 1 or an updated binary restart file. This feature is described in the next two sections.

```
************************
* initial guess options *
************************
 1 = numerical integration
 2 = binary data file
---------------------
1
```

If the user elects to use a binary data file (option 2 above) for the initial guess, the following text input specifies the name of the data file to use.

```
name of binary initial guess data file
sts_cr.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess option 1, the software will create a binary restart file. For initial guess option 2, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*****************************
* binary restart file option *
*****************************

create/update binary data file (yes or no)
no
```

This next input specifies the type of solution data file to create.

```
****************************************
type of comma-delimited solution data file
****************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
10.0
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
discretization/collocation method
*******************************
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next integer defines the number of initial grid points to use in the collocation modeling of the reentry trajectory.

```
number of initial guess grid points to use
25
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
sts_cr.csv
```

The next series of program inputs are user-defined algorithm control options and parameters for the *Sparse Optimization Suite*.

```
***************************
algorithm control parameters
***************************
```

This input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
100000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input.

```
***************************
sparse NLP iteration output
***************************
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the *Sparse Optimization Suite* optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
**********************
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
***************************
differential equation output
***************************
 1 = none
 2 = terse
```

```
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string no.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of inputs allow the reading and writing of configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `reentry_sos` software. Please consult *Appendix J – Sparse Optimization Suite Configuration File* for additional information about these files.

```
************************************
* optimal control configuration options
************************************

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
sts_cr_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
sts_cr_config1.txt
```

Atmosphere model and constants data file

The software also requires a simple ascii data file named `tutility.dat` that defines the fundamental constants to use during the simulation. The following is the companion data file for this example. If the user provides a value of zero for the Earth's rotation rate, the simulation results will be with respect to a non-rotating Earth.

```
*************************
simulation environment and
planet/utility constants
*************************

radius of the earth (feet)
20925656.8d0

gravitational constant of the earth (feet**3/second**2)
1.407644381252d16

surface gravity (feet/second**2)
32.174d0

earth rotation rate (radians/second)
7.2921151467d-5
```

```
atmospheric surface density (slugs/feet**3)
0.0023765d0

j2 gravity coefficient (non-dimensional)
0.0d0
```

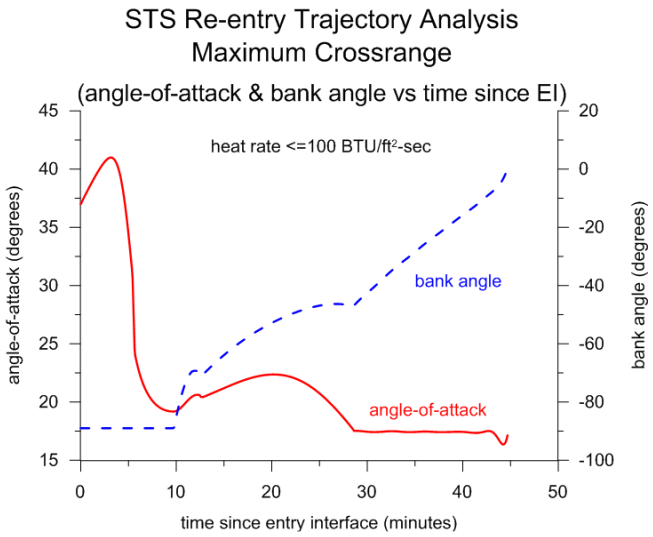
The user's input for `j2` controls the type of gravity model used during the simulation.

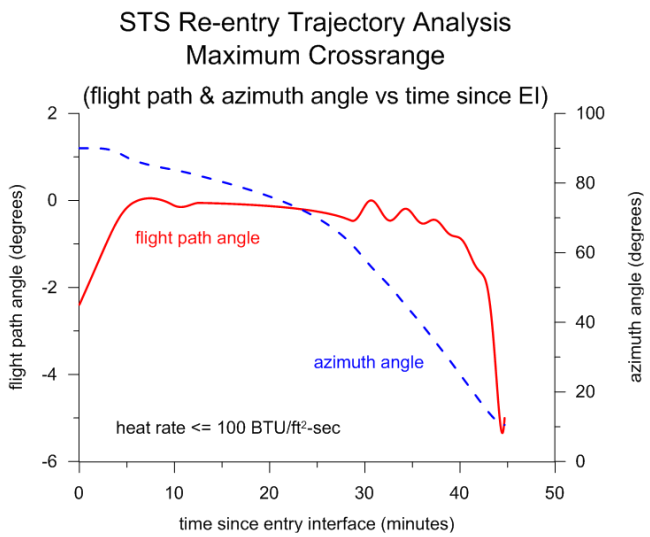
Optimal control solution

The following is the program output for this example.

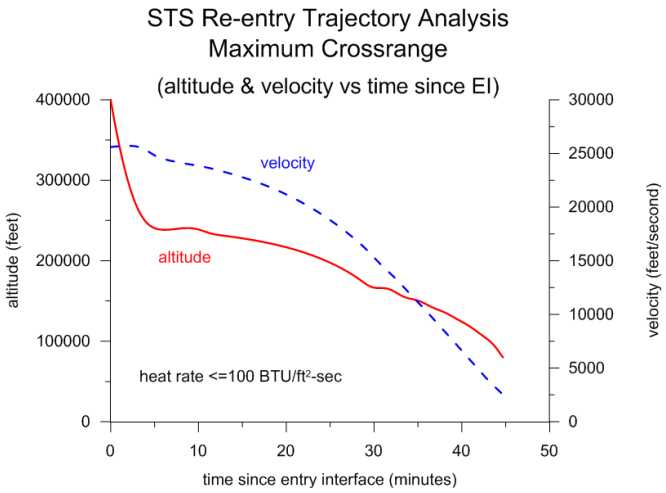
```
program reentry_sos
===================

input file ==> sts_cr.in

maximize crossrange

initial flight path coordinates
-------------------------------

mission elapsed time     00:00:00.000

altitude              400000.000000000       feet
velocity               25600.0000000000       feet/second
declination          1.961736132466737E-032  degrees
longitude            0.000000000000000E+000  degrees
azimuth                90.0000000000000       degrees
flight path angle    -2.39780321520585        degrees

final flight path coordinates
-----------------------------

mission elapsed time     00:44:41.055

altitude               80000.0000000000       feet
velocity               2500.00000000000       feet/second
declination            34.6014881593441       degrees
longitude              120.667074181910       degrees
azimuth                10.5135070714585       degrees
flight path angle     -5.00000000000000       degrees

crossrange             2079.81747860602       nautical miles
downrange              7253.02590513042       nautical miles

heatload               192818.186215868       BTU/ft**2
```

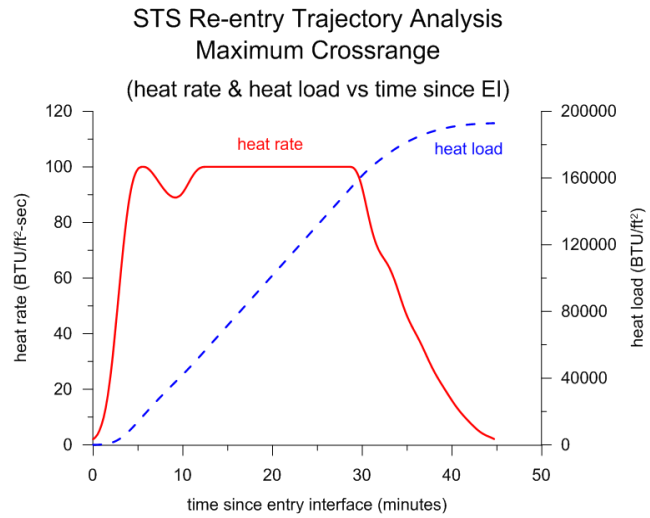
The following are trajectory characteristics plots created from the user-defined summary file. The first two plots illustrate the behavior of the flight controls and the flight path and azimuth angles during the atmospheric reentry.



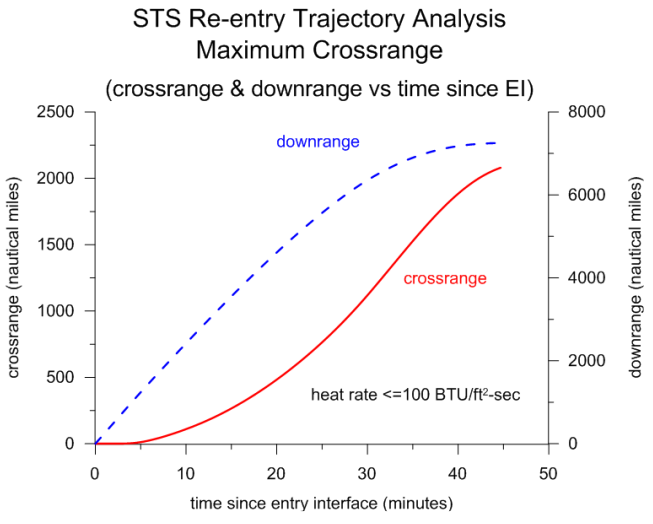


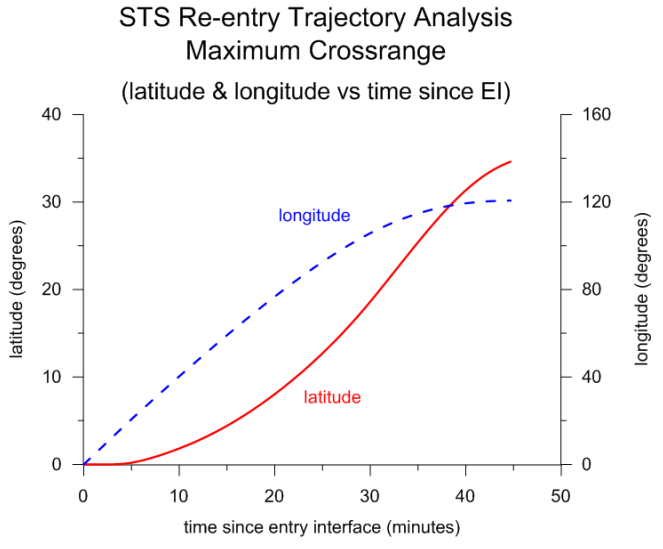
This next two plots summarize the altitude and velocity of the vehicle, and the heat rate and heat load as a function of time since the entry interface (EI).





The final two plots illustrate the behavior of the crossrange, downrange, latitude and longitude of the vehicle during the reentry.





Verification of the optimal control solution

The optimal control solution determined by the software can be verified by numerically integrating the flight path equations of motion with the optimal control solution and the initial entry interface conditions determined by the *SOS* software. This is equivalent to solving an initial value problem (IVP) that uses the optimal control angle-of-attack and bank angle solution.

The following is a display of the final solution computed using this *explicit* numerical integration method.

```
========================================
verification of optimal control solution
========================================

final flight path coordinates
-----------------------------

altitude              79995.7085521678       feet
velocity              2499.73427352356       feet/second
declination           34.6016455556573       degrees
longitude             120.665726280457       degrees
azimuth               10.5094809564327       degrees
flight path angle    -5.00068948061907       degrees

crossrange            2079.82693934372       nautical miles
downrange             7252.94488581160       nautical miles

heatload              192816.843464383       BTU/ft**2
```

Creating an initial guess

The `reentry_sos` computer program creates a *linear* initial guess for the flight parameters and controls. It does this using the initial and final values for the flight path coordinates, angle of attack and bank angles provided by the user. The number of grid points in this initial guess is also provided by the user in the simulation definition data file.

Problem setup

This section provides additional details about the `reentry_sos` software implementation. It explains such things as path constraints and the performance index options.

Performance index

For the *maximum cross-range and downrange* program options, the performance index is set equal to the predicted cross-range or downrange angles at the end of the simulation. These items are computed as final point constraints in the `odepf` subroutine required by the *Sparse Optimization Suite*.

The cross-range angle is determined from the following expression which can be derived from spherical trigonometry relationships.

$$\sin\nu = -\sin\psi_1 \sin\phi_2 \cos\phi_1 \cos\Delta\lambda - \cos\psi_1 \cos\phi_1 \sin\Delta\lambda + \sin\psi_1 \cos\phi_2 \sin\phi_1$$

The downrange angle is determined from the following three equations:

$$\sin\mu = -\cos\psi_1 \sin\phi_2 \cos\phi_1 \cos\Delta\lambda + \sin\psi_1 \cos\phi_1 \sin\Delta\lambda + \cos\psi_1 \cos\phi_2 \sin\phi_1$$

$$\cos\mu = \cos\phi_2 \cos\phi_1 \cos\Delta\lambda + \sin\phi_2 \sin\phi_1$$

$$\mu = \tan^{-1}(\sin\mu, \cos\mu)$$

where

$\phi_1 =$ geocentric declination of the initial point

$\psi_1 =$ flight azimuth at the initial point

$\phi_2 =$ geocentric declination of the final point

$\Delta\lambda = \lambda_2 - \lambda_1$

$\lambda_1 =$ east longitude of the initial point

$\lambda_2 =$ east longitude of the final point

Please note that the inverse tangent above is a four-quadrant calculation.

Path constraints

This section summarizes how the software computes the heat rate and load factor trajectory path constraints.

Heat rate

$$\dot{q} \le \dot{q}_{\max}$$

where $\dot{q}_{\max}$ is the user-defined value of the maximum heat rate.

Load factor

$$n \le n_{\max}$$

where $n_{\max}$ is the user-defined value of the maximum load factor. The instantaneous load factor is determined using the following expression

$$n = \frac{\sqrt{L^2 + D^2}}{W}$$

where L is the lift force, D is the drag force and W is the vehicle weight.

Technical discussion

This scientific simulation models the re-entry trajectory using flight path equations of motion. The equations of motion in this system are summarized in *Appendix B – Trajectory Modeling in the Flight Path System.* This appendix also includes important transformations between Earth-centered-inertial (ECI) and flight path coordinates. The geodetic coordinates used in this simulation can be found in *Appendix G – Aerospace Trajectory Coordinates and Time Systems*.

Aerodynamic characteristics

The aerodynamic model implemented in this software is a simple curve fit of the lift and drag coefficients of the Space Shuttle as a function of angle-of-attack. The lift and drag coefficients for this vehicle model are

$$C_L = C_{L_0} + C_{L_1}\alpha$$

$$C_D = C_{D_0} + C_{D_1}\alpha + C_{D_2}\alpha^2$$

where α is the aerodynamic angle-of-attack.

Heat rate and heat load

The current value of the heat rate is computed using Chapman's stagnation point heat rate equation according to

$$\dot{q} = \frac{dq}{dt} = \frac{17{,}600}{\sqrt{R_N}}\left(\frac{V}{V_0}\right)^{3.15}\sqrt{\frac{\rho}{\rho_0}} \qquad \left(\frac{\text{BTU}}{\text{ft}^2 - \text{sec}}\right)$$

where

$R_N =$ nose radius (feet)

$V =$ relative velocity at the spacecraft location (feet/second)

$V_0 =$ "local circular velocity" at the Earth's surface $= \sqrt{\mu / r_e}$ (feet/second)

$\rho =$ atmospheric density at the spacecraft location (slugs/feet^3)

$\rho_0 =$ atmospheric density at the Earth's surface (slugs/feet^3)

$\mu =$ gravitational constant of the Earth (feet^3/second^2)

$r_e =$ radius of the Earth (feet)

The heat load at any mission elapsed time is determined from $q_L = \int_{t_1}^{t_2} \dot{q}(t)\,dt$.

Atmosphere model

The `reentry_sos` computer program uses a U. S. Standard 1976 atmosphere model.

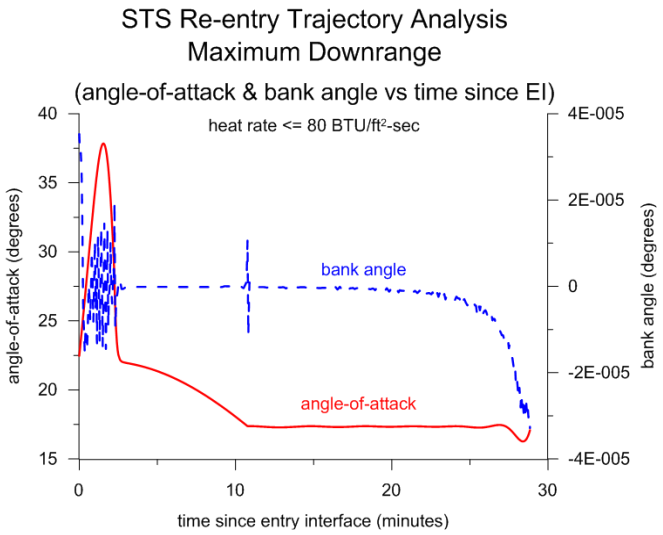
Maximum downrange example

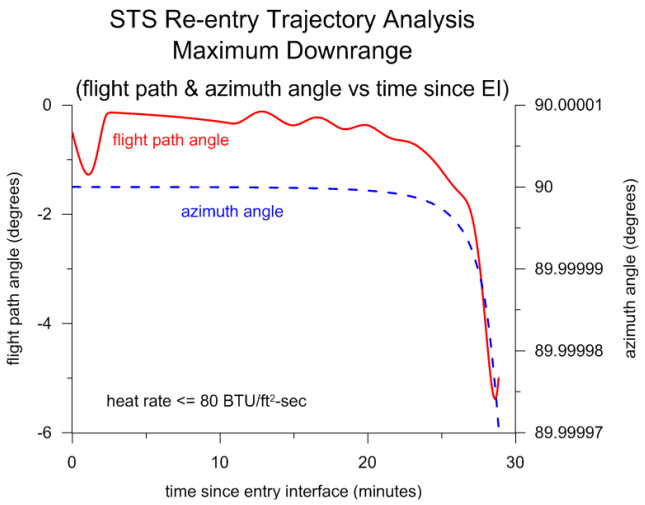
This section summarizes the optimal control solution for a typical maximum downrange example. In this simulation, the initial altitude is 260,000 feet and the initial speed is 20,600 feet/second. The bank angle was bounded according to $-89^\circ \le \psi \le +89^\circ$ and the heat rate mission constraint was 80 BTU/ft^2-second. The vehicle aerodynamics and mass properties are the same as the previous example.

Here is the numerical information for this example.

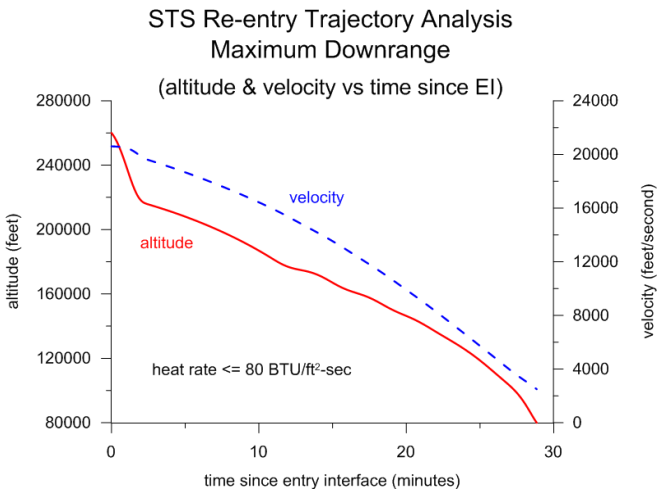
```
program reentry_sos
===================

input file ==> sts_dr.in

maximize downrange

initial flight path coordinates
-------------------------------

mission elapsed time     00:00:00.000

altitude              260000.000000000       feet
velocity               20600.0000000000      feet/second
declination          3.269560220777895E-033  degrees
longitude           -7.362395095980362E-018  degrees
azimuth                90.0000000000000      degrees
flight path angle    -0.500000000000000      degrees

final flight path coordinates
-----------------------------

mission elapsed time     00:28:50.607

altitude               80000.0000000000      feet
velocity               2500.00000000000      feet/second
declination          5.461860341152195E-007  degrees
longitude              61.1239103080595      degrees
azimuth                89.9999704230757      degrees
flight path angle     -5.00000000000000      degrees

crossrange           3.283001283189407E-005  nautical miles
downrange              3674.02050553478      nautical miles

heatload               89847.1411201463      BTU/ft**2

=======================================
verification of optimal control solution
=======================================

final flight path coordinates
-----------------------------

altitude               80000.0227763723      feet
velocity               2500.00079279145      feet/second
declination          5.461755180384940E-007  degrees
longitude              61.1239135596075      degrees
azimuth                89.9999704223379      degrees
flight path angle     -4.99999729726449      degrees

crossrange           3.282938073422019E-005  nautical miles
downrange              3674.02070097800      nautical miles

heatload               89847.1498568508      BTU/ft**2
```

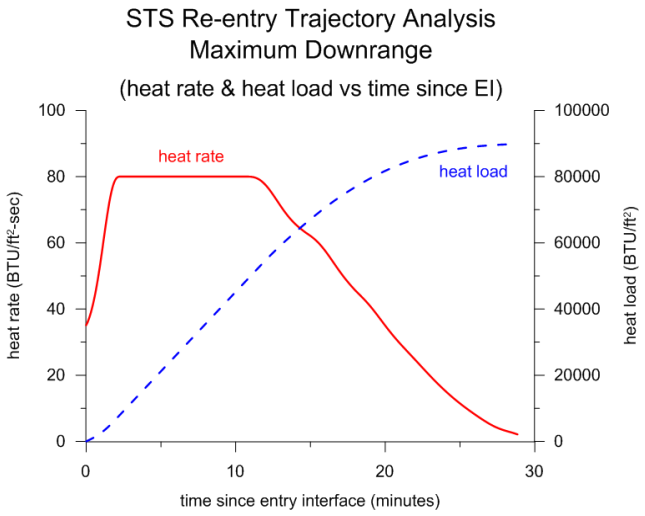
The following are graphic displays of the important control and trajectory characteristics for this maximum downrange example.

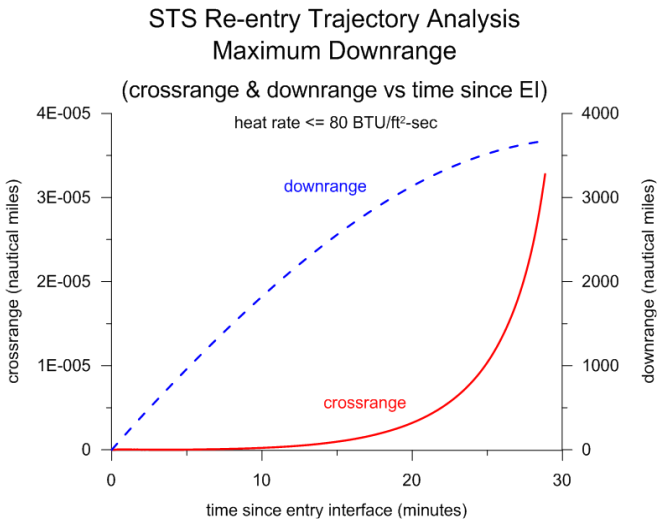


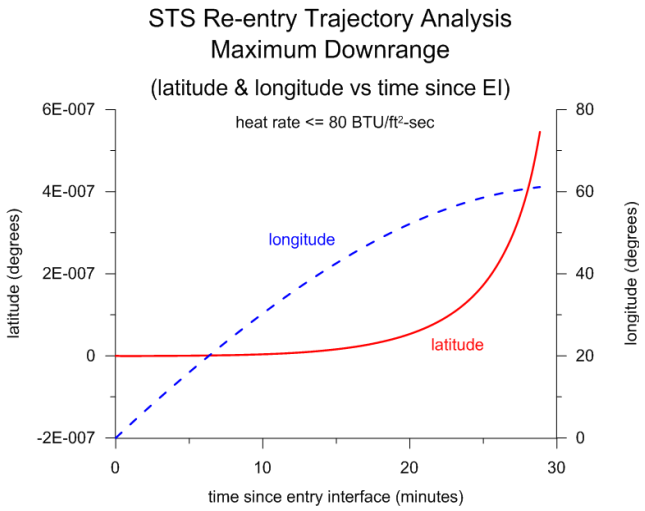


As expected, there is little change in the bank and azimuth angles during the reentry.



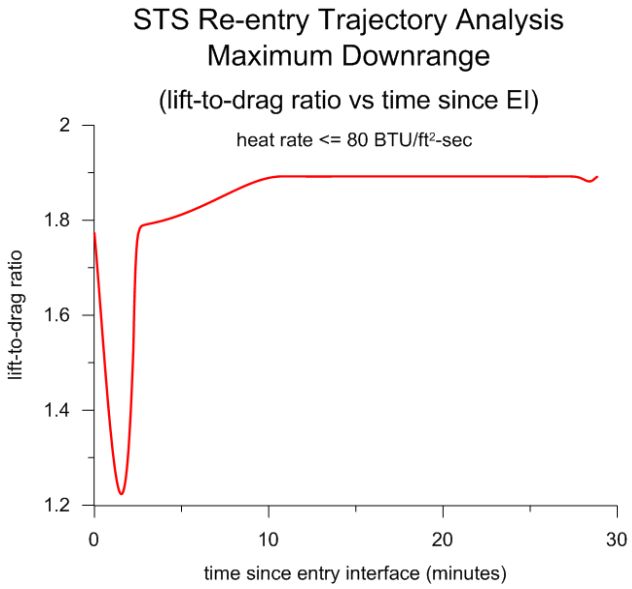






As expected, there is little change in both the cross-range distance and latitude during the mission.

This final plot illustrates the fact that after satisfying the heat rate mission constraint, the vehicle wants to fly at maximum lift-to-drag (L/D) for this maximum downrange mission.



Contents of the simulation summary csv file

This section is a summary of the information contained in the CSV data file produced by the `reentry_sos` software. The user-defined, comma-separated-variable disk file is created by the `odeprt` Fortran subroutine and contains the following information.

time (min) = simulation time since entry interface in minutes

altitude (ft) = altitude relative to a spherical Earth in feet

velocity (fps) = Earth-relative velocity in feet per second

flight path angle (d) = Earth-relative flight path angle in degrees

azimuth (deg) = Earth-relative azimuth angle in degrees

declination (deg) = geocentric declination in degrees

longitude (deg) = geographic longitude in degrees

mach number = Mach number (non-dimensional)

dynamic pressure (psf) = dynamic pressure in pounds per square foot

angle of attack (deg) = angle-of-attack in degrees

bank angle (deg) = bank angle in degrees

heat rate (btu/ft^2-s) = heat rate in BTU/ft^2-second

heat load (btu/ft^2) = accumulated heat load in BTU/ft^2

load factor (g) = normal load factor in g's

lift-to-drag = lift-to-drag ratio (non-dimensional)

lift coefficient = lift coefficient (non-dimensional)

drag coefficient = drag coefficient (non-dimensional)

lift force (lbf) = lift force in pounds

drag force (lbf) = drag force in pounds

density (slugs/ft^3) = atmospheric density in slugs per cubic feet

pressure (psf) = atmospheric pressure in pounds per square foot

temperature (deg K) = atmospheric temperature in degrees K

crossrange (nm) = cross-range distance in nautical miles

downrange (nm) = downrange distance in nautical miles

altitude rate (fps) = rate of change of altitude in feet per second

longitude rate (dps) = rate of change of longitude in degrees per second

declination rate (dps) = rate of change of declination in degrees per second

velocity rate (fps/s) = rate of change of velocity in feet per second per second

fpa rate (dps) = rate of change of flight path angle in degrees per second

azimuth rate (dps) = rate of change of azimuth angle in degrees per second

The accumulated heat load is determined from a cubic spline integration of the heat rate at all collocation nodes of the converged, optimized trajectory. The rate of change of the flight variables is determined from the flight path equations of motion.

Optimal Trajectories in Atmospheric Flight, N. X. Vinh, Elsevier, 1981.

Hypersonic and Planetary Flight Mechanics, N. X. Vinh, R. D. Culp, and A. Busemann, University of Michigan Press, 1980.

"Solving the Optimal Control Problem Using a Nonlinear Programming Technique Part 3: Optimal Shuttle Reentry Trajectories", K. Zondervan, T. Bauer, J. Betts and W. Huffman, AIAA 84-2039.

"Optimum Skip Trajectories Over a Rotating Planetary Atmosphere", N. X. Vinh, IAF-98-A.1.03, 49th International Astronautical Congress, Sept 28-Oct 2, 1998.

"Asymptotic Analysis of Quasi-Equilibrium Glide in Lifting Entry Flight", Ping Lu, AIAA 2005-6128, AIAA Atmospheric Flight Mechanics Conference and Exhibit, 15-18 August 2005.

"Hypersonic, Aerodynamically Controlled, Path Constrained Reentry Optimization in SSOS", C. L. Ranieri, AAS 11-242, 2011.

CMATO 6 – Aero-assist Trajectory Optimization

This *CMATO* application is a Fortran computer program named `aeroassist_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the aero-assist trajectory optimization problem. The trajectory is modeled as a single phase with several types of user-defined initial and final boundary conditions. The software attempts to maximize the speed of the vehicle at atmospheric exit or maximize the orbital plane change during the atmospheric phase of the mission. The type of optimization is selected by the user.

The important features of this scientific simulation are as follows:

- normalized lift coefficient and bank angle control variables
- 3-DOF flight path equations of motion relative to a spherical, rotating Earth
- U.S. Standard 1976 atmosphere model and fourth-order zonal gravity model
- user-defined aerodynamic characteristics and point-mass vehicle properties
- user-defined path constraints such as heat-rate and dynamic pressure

The *Sparse Optimization Suite* software suite is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* software can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `aeroassist_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* software will use this information to *automatically* transcribe the user's problem and perform the optimization using a sparse nonlinear programming method.

The software allows the user to select the type of collocation method and other important algorithm control parameters.

Input file format and contents

The `aeroassist_sos` software is "data-driven" by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in times font.

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly.

The following data file defines a simulation that maximizes the speed at atmospheric exit while enforcing a heat rate path constraint. The simulation starts in a geosynchronous equatorial orbit (GEO) and finishes in an elliptical low Earth orbit (LEO) with a final orbital inclination of 28.5 degrees. The heat-rate path constraint enforces a value <= 600 BTU/foot^2-second.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
******************************************************
aeroassist trajectory optimization
simulation definition data file ==> geo2leo_max_speed.in
geo-to-leo w/ plane change and aeroheating constraints
maximize speed at atmospheric exit - June 9, 2011
******************************************************
```

The first program input is an integer that defines the type of entry interface conditions to use. Please consult the "Problem setup" section later in this document for an explanation of these three program options.

```
**********************
initial conditions type
**********************
 1 = user input of flight path coordinates at entry interface
 2 = derived from deorbit maneuver; fixed entry conditions
 3 = derived from deorbit maneuver; bounded entry conditions
------------------------------------------------------------
3
```

The next program input is an integer that defines the type of mission to simulate. Please note that program option 0 (no optimization) will solve the two-point boundary value problem (TPBVP).

```
**************
simulation type
**************
 0 = no optimization
 1 = maximize final speed
 2 = maximize inclination change
--------------------------------
1
```

This section allows the user to define the vehicle weight, aerodynamic reference area, the nose radius used in the aero-heating calculations, and other vehicle aerodynamic properties.

```
*******************************************
vehicle weight and aerodynamics characteristics
```

```
*********************************************

vehicle weight (pounds)
5000.0

aerodynamic reference area (square feet)
125.84

nose radius (feet)
1.0

drag coefficient at zero angle-of-attack; cd0 (nondimensional)
0.032

drag polar constant k (non-dimensional)
1.4
```

The next set of inputs defines the conditions at the deorbit point in the initial circular orbit. The calendar date and universal time are required in order to transform coordinates.

```
******************
deorbit conditions
******************

calendar date at deorbit maneuver (month, day, year)
1,1,2001

universal time at deorbit maneuver (hours, minutes, seconds)
0,0,0

altitude at deorbit maneuver (nautical miles)
19323.0

orbital inclination at deorbit maneuver (degrees)
0.0

right ascension of the ascending node at deorbit maneuver (degrees)
0.0

true anomaly at deorbit maneuver (degrees)
30.0
```

The following series of data items are reserved for the initial conditions at the entry interface (EI) or point of atmospheric entry. To constrain one or more initial conditions, the user should input identical lower and upper bounds. To free or un-constrain one or more initial states, set the lower and/or upper bounds to `1.0d99`. Please note the units and valid data range for each item.

```
*********************************************
flight conditions and bounds at atmospheric entry
*********************************************
NOTE 1: set upper and lower bounds to
the initial value to constrain or
"fix" a flight condition.
NOTE 2: set bound to 1.0d99 to ignore
-----------------------------------

calendar date at entry interface (month, day, year)
1,1,2001

universal time at entry interface (hours, minutes, seconds)
0,0,0

initial time (seconds)
0.0
```

```
upper bound for initial time (seconds)
0.0

lower bound for initial time (seconds)
0.0

initial altitude (feet)
400000.0

upper bound for initial altitude (feet)
400000.0

lower bound for initial altitude (feet)
400000.0

initial velocity (feet/second)
32268.637

upper bound for initial velocity (feet/second)
35000.0

lower bound for initial velocity (feet/second)
20000.0

initial flight path angle (-90 <= fpa <= +90; degrees)
-4.0

upper bound for initial flight path angle (-90 <= fpa <= +90; degrees)
-0.5

lower bound for initial flight path angle (-90 <= fpa <= +90; degrees)
-10.0

initial flight azimuth (0 <= azimuth <= 360; degrees)
94.0

upper bound for initial flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

lower bound for initial flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

initial declination (-90 <= declination <= +90; degrees)
0.0

upper bound for initial declination (-90 <= declination <= +90; degrees)
1.0d99

lower bound for initial declination (-90 <= declination <= +90; degrees)
1.0d99

initial east longitude (0 <= longitude <= 360; degrees)
18.0

upper bound for initial east longitude (0 <= longitude <= 360; degrees)
0.0

lower bound for initial east longitude (0 <= longitude <= 360; degrees)
0.0
```

The following series of data items allow the user to define the flight conditions at atmospheric exit. To constrain one or more conditions, the user should input identical lower and upper bounds. To free or un-constrain one or more final states, set the lower and/or upper bounds to `1.0d99`. Please note the units and valid data range for each item.

```
***********************************************
flight conditions and bounds at atmospheric exit
***********************************************
NOTE 1: set upper and lower bounds
to the final value to constrain or
"fix" a flight condition.
NOTE 2: set bound to 1.0d99 to ignore
-------------------------------------

final time (seconds)
750.0

upper bound for final time (seconds)
10000.0

lower bound for final time (seconds)
100.0

final altitude (feet)
400000.0

upper bound for final altitude (feet)
400000.0

lower bound for final altitude (feet)
400000.0

final velocity (feet/second)
24314.43

upper bound for final velocity (feet/second)
35000.0

lower bound for final velocity (feet/second)
1000.0

final flight path angle (-90 <= fpa <= +90; degrees)
1.25

upper bound for final flight path angle (-90 <= fpa <= +90; degrees)
+20.0

lower bound for final flight path angle (-90 <= fpa <= +90; degrees)
-20.0

final flight azimuth (0 <= azimuth <= 360; degrees)
90.0

upper bound for final flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

lower bound for final flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

final declination (-90 <= declination <= +90; degrees)
+10.0

upper bound for final declination (-90 <= declination <= +90; degrees)
1.0d99

lower bound for final declination (-90 <= declination <= +90; degrees)
1.0d99

final east longitude (0 <= longitude <= 360; degrees)
30.0
```

```
upper bound for final east longitude (0 <= longitude <= 360; degrees)
1.0d99

lower bound for final east longitude (0 <= longitude <= 360; degrees)
1.0d99
```

The next series of data inputs define lower and upper bounds on the state variables during the aero-assist phase. To free or un-constrain one or more states, set the lower and/or upper bounds to `1.0d99`.

```
************************************************************
upper and lower bounds on the flight conditions during phase
************************************************************
NOTE: set bound to 1.0d99 to ignore
-----------------------------------

upper bound for altitude (feet)
450000.0d0
lower bound for altitude (feet)
10000.0

upper bound for velocity (feet/second)
35000.0d0

lower bound for velocity (feet/second)
1000.0

upper bound for flight path angle (-90 <= fpa <= +90; degrees)
+20.0

lower bound for flight path angle (-90 <= fpa <= +90; degrees)
-20.0

upper bound for flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

lower bound for flight azimuth (0 <= azimuth <= 360; degrees)
1.0d99

upper bound for declination (-90 <= declination <= +90; degrees)
1.0d99

lower bound for declination (-90 <= declination <= +90; degrees)
1.0d99

upper bound for east longitude (0 <= longitude <= 360; degrees)
1.0d99

lower bound for east longitude (0 <= longitude <= 360; degrees)
1.0d99
```

This section of the input file defines the initial guesses and bounds for the control variables. To free or un-constrain one or more control variables, set the lower and/or upper bounds to `1.0d99`.

```
********************************
initial flight controls and bounds
********************************
NOTE 1: set upper and lower bounds
to the initial value to constrain
or "fix" a flight control.
NOTE 2: set bound to 1.0d99 to ignore
-------------------------------------

initial normalized lift coefficient
0.5
```

```
upper bound for initial normalized lift coefficient
+2.0

lower bound for initial normalized lift coefficient
+0.0d0

initial bank angle (degrees)
-90.0d0

upper bound for initial bank angle (degrees)
+0.0d0

lower bound for initial bank angle (degrees)
-180.0d0
```

This section of the input file defines the final guesses and bounds for the control variables. To free one or more control variables, set the lower and/or upper bounds to 1.0d99.

```
*******************************
final flight controls and bounds
*******************************
NOTE 1: set upper and lower bounds
to the final value to constrain
or "fix" a flight control.
NOTE 2: set bound to 1.0d99 to ignore
-------------------------------------

final normalized lift coefficient
0.5

upper bound for final normalized lift coefficient
+2.0

lower bound for final normalized lift coefficient
+0.0

final bank angle (degrees)
-90.0d0

upper bound for final bank angle (degrees)
+0.0d0

lower bound for final bank angle (degrees)
-180.0d0
```

This section of the input file defines the bounds for the control variables during the aero-assist phase. To free or un-constrain one or more control variables, set the lower and/or upper bounds to `1.0d99`.

```
*********************************************************
upper and lower bounds on the flight controls during phase
*********************************************************
NOTE: set bound to 1.0d99 to ignore
-----------------------------------

upper bound for normalized lift coefficient
+2.0d0

lower bound for normalized lift coefficient
+0.0d0

upper bound for bank angle (degrees)
+0.0d0

lower bound for bank angle (degrees)
-180.0d0
```

This next section allows the user to define and enforce one or more point and path constraints during the atmospheric phase. Path constraints are enforced at all points along the trajectory and point constraints are enforced at atmospheric exit only. The user should be careful not to enforce constraints that are inconsistent with either the initial and/or final boundary conditions. For example, while maximizing the final orbital inclination do not enforce an orbital inclination point constraint.

```
******************
flight constraints
******************

enforce an orbital inclination constraint (yes or no)
yes

orbital inclination constraint value (degrees)
28.5d0

enforce a heating rate constraint (yes or no)
yes

heating rate upper bound constraint value (BTU/foot**2-second)
600.0d0

enforce a dynamic pressure constraint (yes or no)
no

dynamic pressure upper bound constraint value (pounds per square foot)
700.0d0
```

The type of trajectory initial guess or restart is specified by the next integer input. Program option 1 will use a simple linear initial guess created from the initial and final values provided by the user. Option 2 will read and use a binary file to initialize the simulation. Be sure to create a binary file first.

```
****************************
initial guess/restart option
****************************
 1 = linear guess
 2 = binary data file
---------------------
1
```

If the user elects to use a binary data file (option 2 above) for the initial guess, the following text input specifies the name of the file to use.

```
name of binary restart file
geo2leo_max_speed.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess options 1, the software will create a binary restart file. For initial guess option 2, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*************************
binary restart file option
*************************

create/update binary restart file (yes or no)
no
```

This next input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by the

Sparse Optimization Suite. Options 2 and 3 allow the user to specify either the number of nodes (option 2) or time step size of the data file (option 3).

```
****************************************
type of comma-delimited solution data file
****************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this next input defines either the number of data points (option 2) or the time step size of the data output in the solution file (option 3).

```
number of user-defined nodes or print step size in solution data file
10.0
```

The software also creates a comma-separated-variable (csv) ASCII data file that contains the optimal control solution and many other flight parameters. The name of this output file is specified in the next line of information.

```
name of solution output file
geo2leo_max_speed.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
discretization/collocation method
*******************************
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

This input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
100000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
**************************
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
**********************
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
***************************
differential equation output
***************************
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* software user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
******************
a0b0c0d0e0f0g0h0i0j1k0l0m0n0o0p0q0r0
```

The last series of inputs allow the reading and writing of configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `aeroassist_sos` software. Please consult *Appendix J – Sparse Optimization Suite Configuration File* for information about this file.

```
************************************
* optimal control configuration options
************************************
```

```
read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
aeroassist_config1.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
aeroassist_config1.txt
```

Atmosphere model and constants data file

The `aeroassist_sos` software also requires a user-created ASCII data file named `tutility.dat` that defines the fundamental constants that will be used during the simulation. The following is the companion data file for this example. If the user provides a value of zero for the Earth's rotation rate, the simulation results will be with respect to a non-rotating, spherical Earth.

```
*************************
simulation environment and
planet/utility constants
*************************

radius of the earth (feet)
20925656.8d0

gravitational constant of the earth (feet**3/second**2)
1.407644381252d16

surface gravity (feet/second**2)
32.174d0

earth rotation rate (radians/second)
7.2921151467d-5

atmospheric surface density (slugs/feet**3)
0.0023765d0

j2 gravity coefficient (non-dimensional)
1.08262668355d-3

j3 gravity coefficient (non-dimensional)
0.0d0

j4 gravity coefficient (non-dimensional)
0.0d0
```

The user's input for `j2`, `j3` and `j4` control the type of gravity model used during the simulation.

Optimal control solution and graphics

After the `aeroassist_sos` scientific simulation has converged, it will display a summary of the initial conditions and the optimized trajectory. It also provides a summary of the relative flight path coordinates and the aerodynamic characteristics of the vehicle. The classical orbital elements at atmospheric exit are determined from the inertial state vector which is computed using the relative flight path coordinates at exit.

The following is the solution summary for this example.

program aeroassist_sos
==========

input file ==> geo2leo_max_speed.in

bounded entry conditions derived from deorbit maneuver

maximize speed at atmospheric exit

orbital elements and state vector prior to deorbit impulse
---

calendar date January 1, 2001

universal time 00:00:00.000

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.227669201902D+05	0.111022302463D-15	0.000000000000D+00	0.000000000000D+00

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.000000000000D+00	0.300000000000D+02	0.300000000000D+02	0.143607658003D+04

r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.227669201902D+05	0.193230000000D+05	0.227669201902D+05	0.193230000000D+05

rx (ft)	ry (ft)	rz (ft)	rmag (ft)
0.119801136085D+09	0.691672181680D+08	0.000000000000D+00	0.138334436336D+09

vx (fps)	vy (fps)	vz (fps)	vmag (fps)
-.504372416457D+04	0.873598651240D+04	0.000000000000D+00	0.100874483291D+05

orbital elements and state vector after deorbit impulse
---

calendar date January 1, 2001

universal time 00:00:00.000

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.131213107012D+05	0.735110211827D+00	0.000000000000D+00	0.210000000000D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.000000000000D+00	0.180000000000D+03	0.300000000000D+02	0.628328849946D+03

r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.347570121219D+04	0.317810220239D+02	0.227669201902D+05	0.193230000000D+05

rx (ft)	ry (ft)	rz (ft)	rmag (ft)
0.119801136085D+09	0.691672181680D+08	0.000000000000D+00	0.138334436336D+09

vx (fps)	vy (fps)	vz (fps)	vmag (fps)
-.259587595393D+04	0.449618904236D+04	0.000000000000D+00	0.519175190787D+04

flight path coordinates at atmospheric entry
---

altitude	400000.000000000	feet
velocity	32268.5194754447	feet/second
declination	2.637383834618052E-013	degrees
longitude	18.8080427045337	degrees
azimuth	90.0000000000005	degrees
flight path angle	-5.45162080704044	degrees
inertial fpa	-5.20130119814137	degrees

orbital elements and state vector at atmospheric entry

calendar date January 1, 2001

universal time 05:11:55.842

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.131213104261D+05	0.735110206972D+00	0.581675704578D-12	0.210000003999D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.000000000000D+00	0.347714838486D+03	0.197714842485D+03	0.628328830184D+03

r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.347570120302D+04	0.317810128543D+02	0.227669196491D+05	0.193229994589D+05

rx (ft)	ry (ft)	rz (ft)	rmag (ft)
-.203144515089D+08	-.648896739412D+07	0.981641981048D-07	0.213256568000D+08

vx (fps)	vy (fps)	vz (fps)	vmag (fps)
0.131677427163D+05	-.311479253279D+05	-.318848277882D-09	0.338168996284D+05

flight path coordinates at atmospheric exit

altitude	400000.000000000	feet
velocity	25600.1592225824	feet/second
declination	11.8749410000429	degrees
longitude	59.7589681113454	degrees
azimuth	62.4006932817757	degrees
flight path angle	2.85290494398532	degrees

orbital elements and state vector at atmospheric exit

calendar date January 1, 2001

universal time 05:21:20.255

sma (nm)	eccentricity	inclination (deg)	argper (deg)
0.390347950906D+04	0.111289864741D+00	0.285000000002D+02	0.357704478388D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (min)
0.218238520146D+03	0.278426340657D+02	0.255471124534D+02	0.101952769539D+03

r-perigee (nm)	h-perigee (nm)	r-apogee (nm)	h-apogee (nm)
0.346906180248D+04	0.251416123121D+02	0.433789721564D+04	0.893977025480D+03

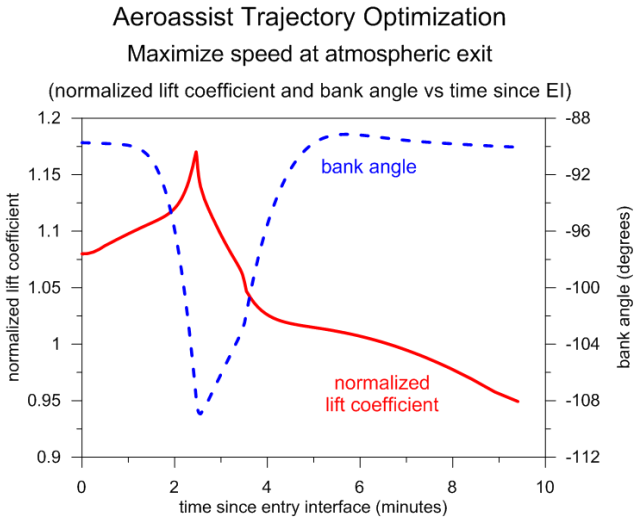
rx (ft)	ry (ft)	rz (ft)	rmag (ft)
-.101099984601D+08	-.182568968034D+08	0.438831268249D+07	0.213256568000D+08

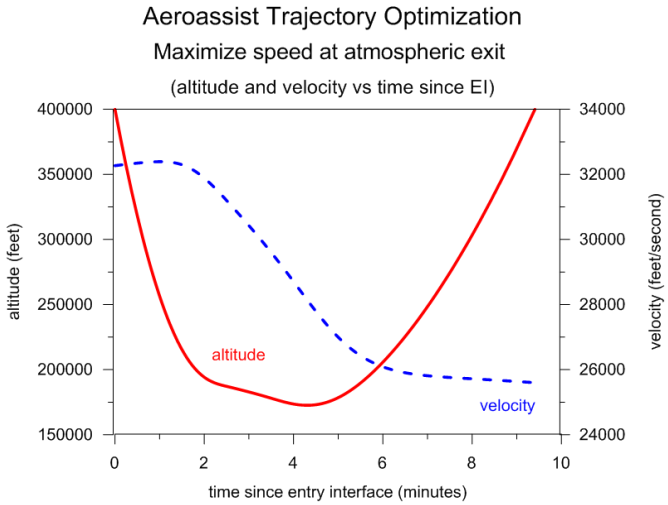
vx (fps)	vy (fps)	vz (fps)	vmag (fps)
0.217306749776D+05	-.106726700692D+05	0.118541683058D+05	0.269564357364D+05

aerodynamic characteristics

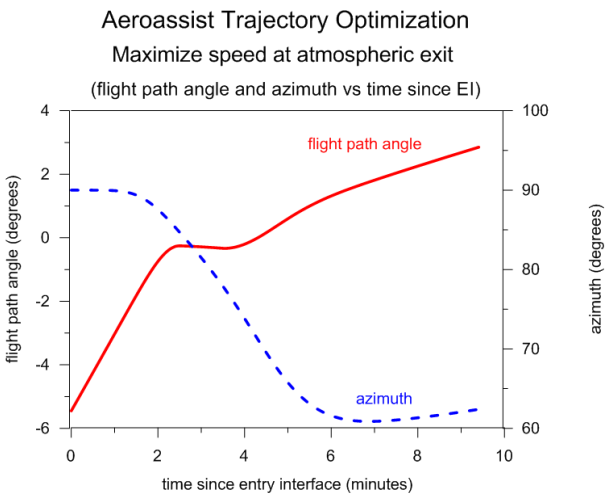
drag coefficient at aoa = 0 degrees	3.200000000000000E-002
drag coefficient at max L/D	6.400000000000000E-002
lift coefficient at max L/D	0.151185789203691
maximum lift-to-drag ratio	2.36227795630767

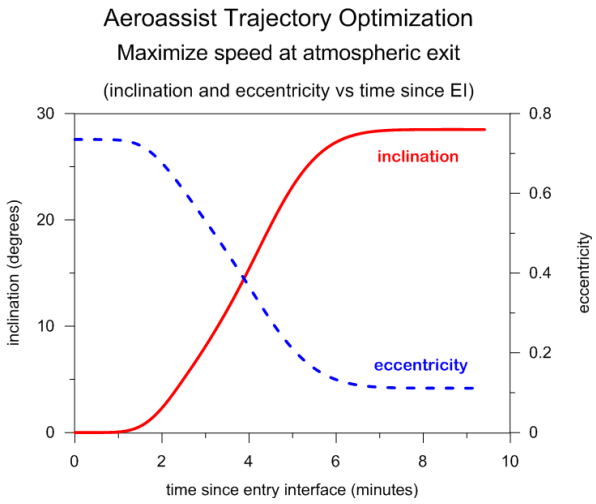
The following are plots created from the trajectory summary file. The first plot shows the behavior of the normalized lift coefficient and bank angle during the atmospheric pass. The second plot summarizes the altitude and relative velocity of the vehicle as a function of time since the entry interface (EI).



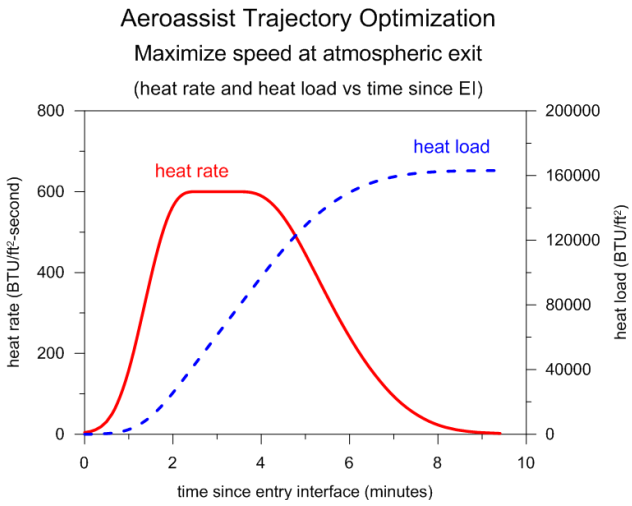


The plot on the left illustrates the behavior of the relative flight path and azimuth angles during the aero-assist pass through the atmosphere. The plot on the right shows the behavior of the orbital eccentricity and inclination during the atmospheric phase of the mission.





The last plot illustrates the behavior of the heat rate and heat load during the atmospheric portion of the trajectory. It confirms that the solution has satisfied the maximum heat-rate path constraint.



Problem setup

This section provides additional details about the *Sparse Optimization Suite* implementation. It briefly explains such things as initial conditions, path constraints and the performance index options.

Entry interface initial conditions

The `aeroassist_sos` computer program includes three options for specifying initial conditions at the entry interface (EI). This section summarizes these options and describes how the user invokes each.

<u>user input of flight path coordinates at entry interface</u>

For this option, all Earth relative coordinates at the entry interface are defined by the user. The user can provide initial guesses and lower and upper bounds for these coordinates in the `flight conditions and bounds at atmospheric entry` part of the input data file.

<u>derived from deorbit maneuver; fixed entry conditions</u>

For this program option, the software uses the flight path angle and entry altitude provided in the input data file to constrain the entry altitude and *inertial* flight path angle. The software then uses the deorbit algorithm described in *Appendix E – Impulsive De-orbit from Earth Orbit* to compute the Earth relative flight path angle, speed and other flight path coordinates at the entry interface. This option is valid for initial circular orbits.

<u>derived from deorbit maneuver; bounded entry conditions</u>

For this program option, the software will use the entry altitude provided in the input data file to constrain the entry altitude. The flight path angle provided in the data file is used for an *inertial* flight path angle initial guess. During the trajectory optimization, the software will change the inertial flight path angle between the lower and upper bounds provided by the user (the inertial flight path angle is treated as a problem parameter). The software then uses the deorbit algorithm described later in this section to compute the Earth relative flight path angle, speed and other coordinates at the entry interface. This option is valid for initial circular orbits.

For the second and third initial conditions options, the `aeroassist_sos` computer program calculates the single impulsive maneuver required to establish an entry interface altitude and flight path angle relative to the user-defined initial circular orbit. The algorithm uses a tangential, impulsive delta-v applied opposite to the velocity vector to establish the deorbit trajectory. The entry altitude and flight path angle initial guesses are provided by the user.

Performance index

This section describes the two types of trajectory optimization performed by the `aeroassist_sos` software.

maximize final speed

The performance index for this type of optimization is simply $J = v_f$ where v_f is the relative speed of the vehicle as it exits from the atmosphere. For this program option, the optimization indicator is set to `maxmin = +1`. This option minimizes the energy loss during the aero-assist maneuver.

maximize inclination change

The performance index for this program option is given by $J = \cos^{-1}\left(h_z/h\right)$ where h_z is the z-component of the ECI angular momentum vector and *h* is the angular momentum magnitude of the vehicle at exit from the atmosphere at the user-defined altitude. For this program option, the optimization indicator is also set to `maxmin = +1`.

Path and point constraints

This section summarizes how the software computes the heat rate and dynamic pressure path constraints, and the orbital inclination point constraint.

dynamic pressure

To enforce this path constraint, the software ensures that $q \le q_{\max}$ where $q_{\max}$ is the user-defined value of the maximum dynamic pressure. The dynamic pressure at any simulation time is given by

$$q = \frac{1}{2}\rho v^2$$

where ρ is the atmospheric density and *v* is the relative speed at the current flight condition.

heat rate

To enforce this path constraint, the software ensures that $\dot{Q} \le \dot{Q}_{\max}$ where $\dot{q}_{\max}$ is the user-defined value of the maximum heat-rate. The heat rate at any simulation time is computed from Chapman's stagnation point equation given by

$$\dot{Q} = \frac{dQ}{dt} = \frac{17,600}{\sqrt{R_N}}\left(\frac{V}{V_0}\right)^{3.15}\sqrt{\frac{\rho}{\rho_0}} \qquad \left(\frac{\text{BTU}}{\text{ft}^2 - \text{sec}}\right)$$

where

R_N = nose radius (feet)

V = relative velocity at the spacecraft location (feet/second)

V_0 = "local circular velocity" at the Earth's surface $= \sqrt{\mu / r_e}$ (feet/second)

ρ = atmospheric density at the spacecraft location (slugs/feet^3)

ρ_0 = atmospheric density at the Earth's surface (slugs/feet^3)

μ = gravitational constant of the Earth (feet^3/second^2)

r_e = radius of the Earth (feet)

orbital inclination

A final orbital inclination point constraint is enforced as $\cos i = h_z / h$ where h_z is the z-component of the angular momentum vector and *h* is the angular momentum magnitude at the atmospheric exit. In this equation, *i* is the user-defined final orbit inclination.

Technical discussion

Trajectory mechanics in this computer program are modeled in the flight path coordinate system. The equations of motion are described in *Appendix B – Trajectory Modeling in the Flight Path System* along with coordinate transformations for converting Earth-centered-inertial (ECI) to and from Earth-centered-fixed (ECF) coordinates. Appendix B also includes a summary of the aerodynamic characteristics of aero-assist vehicles

In the `aeroassist_ocs` computer program, the vehicle aerodynamics are modeled using a parabolic drag polar $(n = 2)$. The normalized lift coefficient is given by $\lambda = C_L / C_L^*$ where C_L is the lift coefficient at any simulation conditions and C_L^* is the lift coefficient corresponding to maximum L/D.

In this computer program, the control variables are the normalized lift coefficient and bank angle.

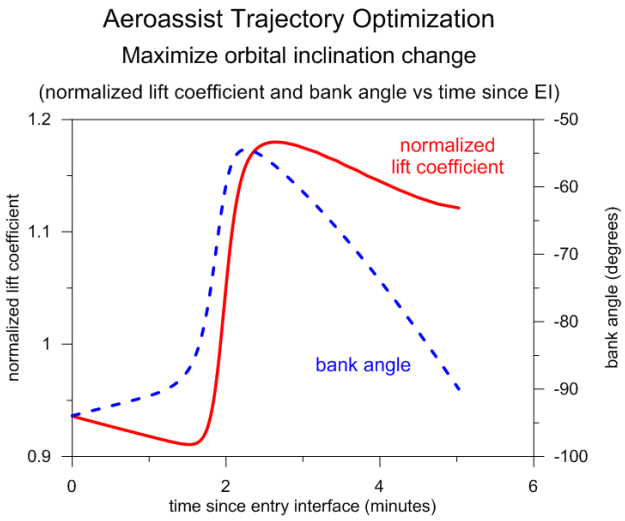
Maximize orbital inclination example

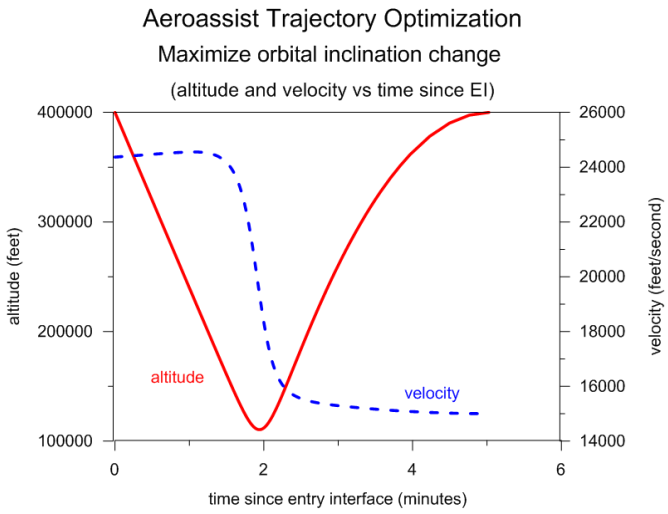
This section presents graphics and a simulation summary for an aero-assist trajectory that maximizes the orbital inclination change. The mission starts in a 500 nautical mile circular Earth orbit with an initial orbital inclination equal to 5 degrees. The speed at atmospheric exit is constrained to be greater than or equal to 15,000 feet per second.

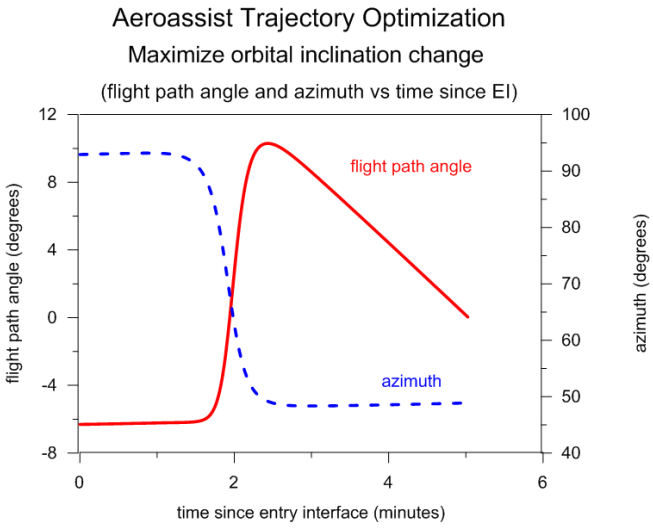
For this type of trajectory optimization make sure the orbital inclination at the atmospheric exit is not constrained by using the following statement in the input file

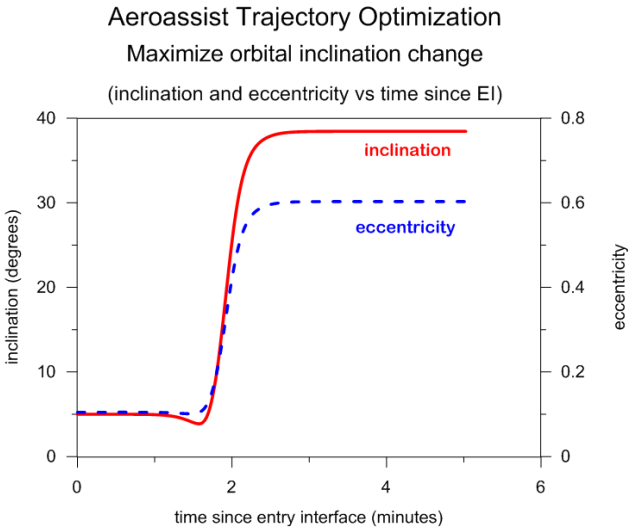
```
enforce an orbital inclination constraint (yes or no)
no
```

The following are plots of the important trajectory parameters for this example.









The following is the `aeroassist_sos` program output for this example.

```
program aeroassist_sos
======================

input file ==> leo2leo_max_inc.in

bounded entry conditions derived from deorbit maneuver

maximize orbital inclination change

orbital elements and state vector prior to deorbit impulse
----------------------------------------------------------

calendar date           January    1, 2001

universal time          00:00:00.000

      sma (nm)              eccentricity       inclination (deg)       argper (deg)
 0.394392019016D+04    0.248334991895D-15    0.500000000000D+01    0.000000000000D+00

     raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.431780632080D-14    0.300000000000D+02    0.300000000000D+02    0.103541236919D+03

   r-perigee (nm)          h-perigee (nm)       r-apogee (nm)          h-apogee (nm)
 0.394392019016D+04    0.500000000000D+03    0.394392019016D+04    0.500000000000D+03
```

```
     rx (ft)                 ry (ft)              rz (ft)              rmag (ft)
 0.207531855633D+08    0.119362626872D+08    0.104428766999D+07    0.239637145430D+08

     vx (fps)              vy (fps)             vz (fps)             vmag (fps)
 -.121182361413D+05    0.209095296884D+05    0.182934680739D+04    0.242364722827D+05

orbital elements and state vector after deorbit impulse
-------------------------------------------------------

calendar date           January    1, 2001

universal time          00:00:00.000

     sma (nm)             eccentricity       inclination (deg)       argper (deg)
 0.357017458724D+04    0.104685525536D+00    0.500000000000D+01    0.210000000000D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.456326147825D-14    0.180000000000D+03    0.300000000000D+02    0.891775123771D+02

   r-perigee (nm)         h-perigee (nm)       r-apogee (nm)         h-apogee (nm)
 0.319642898432D+04    -.247491205844D+03    0.394392019016D+04    0.500000000000D+03

     rx (ft)                 ry (ft)              rz (ft)              rmag (ft)
 0.207531855633D+08    0.119362626872D+08    0.104428766999D+07    0.239637145430D+08

     vx (fps)              vy (fps)             vz (fps)             vmag (fps)
 -.114664033296D+05    0.197848183550D+05    0.173094731598D+04    0.229328066592D+05

flight path coordinates at atmospheric entry
--------------------------------------------

altitude              400000.000000000       feet
velocity              24368.0971648879       feet/second
declination           4.17633335544806       degrees
longitude             16.1785108288942       degrees
azimuth               92.9277990467808       degrees
flight path angle    -6.30689859387635       degrees

inertial fpa         -5.93056753615202       degrees

orbital elements and state vector at atmospheric entry
------------------------------------------------------

calendar date           January    1, 2001

universal time          00:26:03.919

     sma (nm)             eccentricity       inclination (deg)       argper (deg)
 0.357017458725D+04    0.104685525537D+00    0.500000000013D+01    0.209999999997D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.373173703543D-06    0.273322949484D+03    0.123322949481D+03    0.891775123775D+02

   r-perigee (nm)         h-perigee (nm)       r-apogee (nm)         h-apogee (nm)
 0.319642898433D+04    -.247491205835D+03    0.394392019018D+04    0.500000000016D+03

     rx (ft)                 ry (ft)              rz (ft)              rmag (ft)
 -.117154107210D+08    0.177516413688D+08    0.155306738546D+07    0.213256568000D+08

     vx (fps)              vy (fps)             vz (fps)             vmag (fps)
 -.200622216447D+05    -.163311939918D+05    -.142879432473D+04    0.259083401194D+05

flight path coordinates at atmospheric exit
-------------------------------------------

altitude              400000.000000000       feet
velocity              15000.0000000000       feet/second
declination           8.99670339727738       degrees
longitude             29.5623003632225       degrees
azimuth               48.8880628810677       degrees
flight path angle    3.984582451546240E-002  degrees

orbital elements and state vector at atmospheric exit
-----------------------------------------------------
```

```
calendar date           January    1, 2001

universal time          00:31:05.462

      sma (nm)              eccentricity       inclination (deg)       argper (deg)
 0.218954752865D+04    0.602957760689D+00    0.384433270708D+02    0.194591552768D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.126562332360D+03    0.179975688582D+03    0.145672413500D+02    0.428304598571D+02

   r-perigee (nm)         h-perigee (nm)       r-apogee (nm)         h-apogee (nm)
 0.869342853852D+03    -.257457733631D+04    0.350975220345D+04    0.658320132855D+02

      rx (ft)                 ry (ft)              rz (ft)              rmag (ft)
 -.156695378708D+08    0.140757933205D+08    0.333485580891D+07    0.213256568000D+08

      vx (fps)                vy (fps)             vz (fps)             vmag (fps)
 -.743898214339D+04    -.105738521023D+05    0.974327143645D+04    0.161887659164D+05

aerodynamic characteristics
---------------------------

drag coefficient at aoa = 0 degrees   5.000000000000000E-002

drag coefficient at max L/D           0.100000000000000

lift coefficient at max L/D           0.188982236504614

maximum lift-to-drag ratio            1.88982236504614
```

Contents of the simulation summary csv file

This section is a summary of the information contained in the CSV data file produced by the `aeroassist_sos` software. The comma-separated-variable disk file is created by the `odeprt` Fortran subroutine and contains the following information:

time (min) = simulation time since entry interface in minutes

altitude (ft) = altitude relative to a spherical Earth in feet

velocity (fps) = Earth-relative velocity in feet per second

flight path angle (d) = Earth-relative flight path angle in degrees

azimuth (deg) = Earth-relative azimuth angle in degrees

declination (deg) = geocentric declination in degrees

longitude (deg) = geographic longitude in degrees

mach number = Mach number (non-dimensional)

dynamic pressure (psf) = dynamic pressure in pounds per square foot

bank angle (deg) = bank angle in degrees

heat rate (btu/ft^2-s) = heat rate in BTU/ft^2-second

heat load (btu/ft^2) = accumulated heat load in BTU/ft^2

lift-to-drag = lift-to-drag ratio (non-dimensional)

lift coefficient = lift coefficient (non-dimensional)

drag coefficient = drag coefficient (non-dimensional)

lift force (lbf) = lift force in pounds

drag force (lbf) = drag force in pounds

density (slugs/ft^3) = atmospheric density in slugs per cubic feet

pressure (psf) = atmospheric pressure in pounds per square foot

temperature (deg K) = atmospheric temperature in degrees K

crossrange (nm) = cross-range distance in nautical miles

downrange (nm) = downrange distance in nautical miles

altitude rate (fps) = rate of change of altitude in feet per second

longitude rate (dps) = rate of change of longitude in degrees per second

declination rate (dps) = rate of change of declination in degrees per second

velocity rate (fps/s) = rate of change of velocity in feet per second per second

fpa rate (dps) = rate of change of flight path angle in degrees per second

azimuth rate (dps) = rate of change of azimuth angle in degrees per second

perigee altitude (nm) = perigee altitude in nautical miles

perigee radius (nm) = perigee radius in nautical miles

apogee altitude (nm) = apogee altitude in nautical miles

apogee radius (nm) = apogee radius in nautical miles

The accumulated heat load is determined from a cubic spline integration of the heat rate of the optimized solution at all collocation nodes. The rate of change of the flight variables is determined from the equations of motion.

"Minimum-Fuel Aero-assisted Coplanar Orbit Transfer Using Lift Modulation", Kenneth D. Mease and Nguyen X. Vinh, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 8. No. 1, Jan.-Feb. 1985.

"Variational Solutions for the Heat-Rate-Limited Aero-assisted Orbital Transfer Problem", Hans Seywald, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 19, No. 3, May-June, 1996, pp. 686-692.

"Fuel-Optimal Trajectories of Aero-assisted Orbital Transfer with Plane Change", D. S. Naidu, *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 27, No. 2, March 1991, pp. 361-368.

Optimal Trajectories in Atmospheric Flight, N. X. Vinh, Elsevier, 1981.

Hypersonic and Planetary Flight Mechanics, N. X. Vinh, R. D. Culp, and A. Busemann, University of Michigan Press, 1980.

"A Survey of Aero-assisted Orbit Transfer", G. D. Walberg, *AIAA Journal of Spacecraft and Rockets*, Vol. 22, No. 1, Jan-Feb 1985, pp. 3-18.

"Optimal Aero-assisted Return From High Earth Orbit With Plane Change", Nguyen X. Vinh and John M. Hanson, *Acta Astronautica*, Vol. 12, No. 1, 1985.

CMATO 7 – Finite-Burn De-orbit Trajectory Optimization

This *CMATO* application is a Fortran computer program named `deorbit_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the de-orbit trajectory optimization problem. The software models the trajectory as a single, finite-burn propulsive maneuver followed by a user-defined, time-bounded final coast phase. This computer attempts to maximize the final spacecraft mass. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass.

The important features of this scientific simulation are as follows:

- single, continuous thrust orbital maneuver
- variable inertial attitude steering
- constant propulsive thrust magnitude
- modified equinoctial equations of motion with oblate Earth gravity model
- user-specified final coast phase and entry interface (EI) target conditions

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods.

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `deorbit_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method.

The `deorbit_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `deorbit_sos` software is "data-driven" by a user-created text file. This text file should be simple ASCII format with no special characters.

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly.

The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in times font. This example attempts to optimize the maneuver required to de-orbit a spacecraft from a circular Earth orbit (LEO) to typical user-defined entry interface (EI) conditions.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*********************************************
** de-orbit trajectory optimization
** single finite-burn maneuver with final coast
** program deorbit_sos
** cleo2ei.in - May 11, 2012
*********************************************
```

The first two inputs define the calendar date and Universal Coordinated Time (UTC) of the de-orbit maneuver. Please be sure to provide all four digits of the calendar year.

```
maneuver calendar date (month, day, year)
-----------------------------------------
3, 18, 2010

maneuver UTC (hour, minute, second)
-----------------------------------
12, 30, 45.875
```

The next three inputs define the initial mass prior to the propulsive maneuver, and the thrust magnitude and specific impulse of the upper stage or spacecraft propulsion system.

```
initial spacecraft mass (kilograms)
8000.0

thrust magnitude (newtons)
2000.0

specific impulse (seconds)
325.0
```

This next integer input defines the type of initial guess for the propulsive maneuver.

```
***********************************
* type of propulsive initial guess *
***********************************
1 = thrust duration
2 = delta-v
-----------
2
```

The next two numeric inputs define either the user's initial guess for the delta-v magnitude or the maneuver duration and should be consistent with the previous input.

```
initial guess for delta-v (meters/second)
150.0

initial guess for thrust duration (seconds)
700.0
```

The next two inputs define the lower and upper bounds for the thrust duration. These inputs are required for either type of propulsive initial guess.

```
lower bound for thrust duration (seconds)
10.0

upper bound for thrust duration (seconds)
1000.0
```

The next section of the input data file lets the user define the characteristics of a final coast phase that follows the propulsive maneuver. These three inputs define an initial guess for the coast duration as well as lower and upper bounds on the coast duration. All inputs are in minutes.

```
******************
* coast maneuver *
******************

initial guess for coast duration (minutes)
30.0

lower bound for coast duration (minutes)
20.0

upper bound for coast duration (minutes)
40.0
```

The next six inputs define the classical orbital elements of the initial park orbit. These elements are defined with respect to an Earth-centered-inertial (ECI) coordinate system.

```
*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
6878.14d0

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
100.0

right ascension of the ascending node (degrees)
220.0d0

true anomaly (degrees)
180.0
```

This next integer input allows the user to define the type of initial orbit constraints to use during the simulation.

```
***********************************
* initial orbit constraint options *
***********************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-------------------------------------------
3
```

The next series of inputs define the entry interface (EI) mission constraints. These elements are defined with respect to the relative coordinate system. Please note the proper units for each mission constraint. To disregard a mission constraint, input the value `1.0d99` for that constraint.

```
*********************************************************
* entry interface constraints (set to 1.0d99 to ignore) *
*********************************************************

geodetic altitude (kilometers)
121.92d0

relative flight path angle (degrees)
-2.0d0

geodetic latitude (degrees)
-14.0

east longitude (degrees)
181.0

relative azimuth (degrees)
1.0d99

relative velocity (meters/second)
1.0d99
```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will use a J_2 gravity model in the spacecraft equations of motion.

```
************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate gravity model
------------------------
2
```

This next input specifies the type of solution data file to create.

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
2
```

For options 2 or 3, this input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
100
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
cleo2ei.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next user input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the Sparse Optimization Suite NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the *Sparse Optimization Suite* optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the *Sparse Optimization Suite* differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
***************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

Optimal control solution

The following is the optimal control solution for this example. The output includes the time and orbital characteristics at the beginning and end of the propulsive maneuver. This example optimizes the finite-burn maneuver required to transfer from a circular low Earth orbit (LEO) to an entry interface defined by a relative flight path angle, geodetic altitude and latitude, and a geographic east longitude.

```
program deorbit_sos
===================

input data file ==> cleo2ei.in

oblate earth gravity model

initial epoch
-------------

calendar date            March 18, 2010

UTC time                 12:30:45.875

------------------------
beginning of finite burn
------------------------

mission elapsed time     00:00:00.000

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.687814000000D+04    0.256247074642D-15    0.285000000000D+02    0.000000000000D+00
```

```
    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.220000000000D+03    0.199001209386D+03    0.199001209386D+03    0.946163624135D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.371682044841D+04    0.568790088946D+04    -.106856870886D+04    0.687814000000D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.596468785281D+01    0.325246126525D+01    -.343449740362D+01    0.761260651018D+01

------------------
end of finite burn
------------------

mission elapsed time     00:09:39.318

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.662834854917D+04    0.373976098013D-01    0.286374887088D+02    0.314330866858D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.220229462304D+03    0.203823970508D+03    0.235257057194D+03    0.895092125778D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.210335480056D+03    0.629617719700D+04    -.269907170370D+04    0.685354480337D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.711928147802D+01    -.122317403027D+01    -.200086582577D+01    0.749558453521D+01
```

The following program output is the final spacecraft mass, the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v.

```
final mass              7636.46768849416      kilograms

propellant mass         363.532311505839      kilograms

thrust duration         579.318048180925      seconds
                        9.65530080301541      minutes

delta-v                 148.223366147974      meters/second
```

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node or user-defined step size.

This section of the numeric results summarizes the time and orbital conditions at the beginning and end of the final coast.

```
---------------------------
beginning of coast maneuver
---------------------------

mission elapsed time     00:09:39.318

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.662834854917D+04    0.373976098013D-01    0.286374887088D+02    0.314330866858D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.220229462304D+03    0.203823970508D+03    0.235257057194D+03    0.895092125778D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.210335480056D+03    0.629617719700D+04    -.269907170370D+04    0.685354480337D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.711928147802D+01    -.122317403027D+01    -.200086582577D+01    0.749558453521D+01

---------------------
end of coast maneuver
---------------------

mission elapsed time     00:33:37.138

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.663096280332D+04    0.386110555799D-01    0.286533250810D+02    0.307476929735D+02
```

```
   raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
0.220038827173D+03    0.299159749780D+03    0.329907442753D+03    0.895621721271D+02

    rx (km)               ry (km)              rz (km)              rmag (km)
-.614443018434D+04    -.142801717018D+04    -.156247691255D+04    0.649881446349D+04

    vx (kps)              vy (kps)             vz (kps)             vmag (kps)
0.107390419948D+01    -.708739128858D+01    0.334249561358D+01    0.790927698554D+01

coast duration         1437.81997871958      seconds
                       23.9636663119930      minutes
                      0.399394438533216      hours
```

This final section of the numerical results summarizes both the relative and inertial flight conditions at the entry interface.

```
relative flight path coordinates at entry interface
---------------------------------------------------

geodetic altitude    121.919999999994      kilometers

geodetic latitude   -13.9999999999992      degrees

east longitude       180.999999897136      degrees

flight path angle   -2.00000000000018      degrees

azimuth               63.1924684582462      degrees

velocity              7496.23024105710      meters/second

inertial flight path coordinates at entry interface
---------------------------------------------------

right ascension      193.083753391143      degrees

flight path angle   -1.89551468207218      degrees

azimuth               64.6963141835643      degrees

velocity              7909.27698553550      meters/second
```

Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* can be verified by numerically integrating the orbital equations of motion with the SOS-computed initial park orbit conditions and the optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the optimal unit thrust vector solution. This part of the `deorbit_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

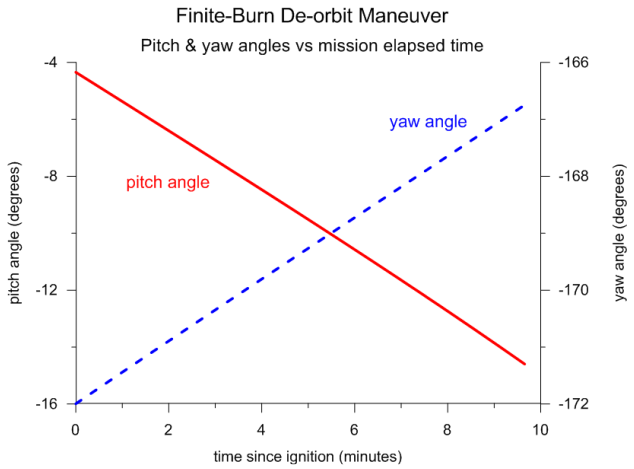
The following is a display of the final solution for the previous example computed using this *explicit* numerical integration method.

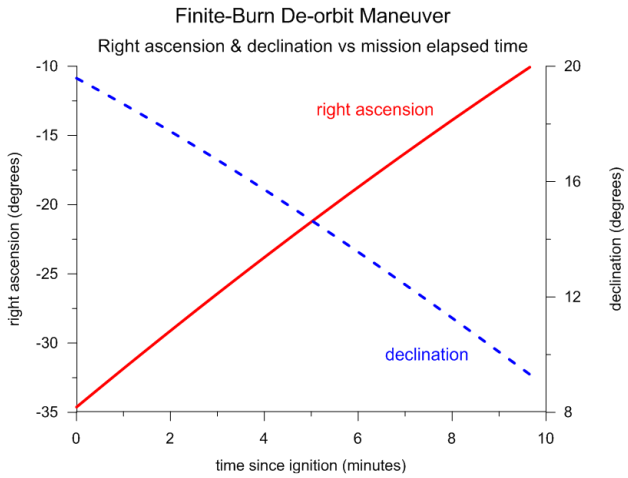
```
=======================================
verification of optimal control solution
=======================================

beginning of coast maneuver
---------------------------

mission elapsed time     00:09:39.318

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.662834854907D+04    0.373976097701D-01    0.286374887086D+02    0.314330866910D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.220229462304D+03    0.203823970504D+03    0.235257057195D+03    0.895092125757D+02
```

```
     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.210335480218D+03    0.629617719672D+04    -.269907170359D+04    0.685354480309D+04

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 -.711928147827D+01    -.122317403037D+01    -.200086582582D+01    0.749558453547D+01

end of coast maneuver
---------------------

mission elapsed time     00:33:37.138

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.663096280315D+04    0.386110556971D-01    0.286533250796D+02    0.307476930541D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.220038827181D+03    0.299159749691D+03    0.329907442745D+03    0.895621721237D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.614443018404D+04    -.142801717004D+04    -.156247691279D+04    0.649881446323D+04

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 0.107390420036D+01    -.708739128871D+01    0.334249561350D+01    0.790927698574D+01

relative flight path coordinates at entry interface
---------------------------------------------------

geodetic altitude    121.919999748011      kilometers

geodetic latitude   -14.0000000027976      degrees

east longitude       180.999999896537      degrees

flight path angle   -2.00000000777524      degrees

azimuth              63.1924684615364      degrees

velocity             7496.23024127568      meters/second

inertial flight path coordinates at entry interface
---------------------------------------------------

right ascension      193.083753390544      degrees

flight path angle   -1.89551468944646      degrees

azimuth              64.6963141865741      degrees

velocity             7909.27698574230      meters/second

mass and propulsive properties
------------------------------

final mass           7636.46768849181      kilograms

propellant mass      363.532311508195      kilograms

thrust duration      579.318048180925      seconds
                     9.65530080301541      minutes

delta-v              148.223363619416      meters/second
```

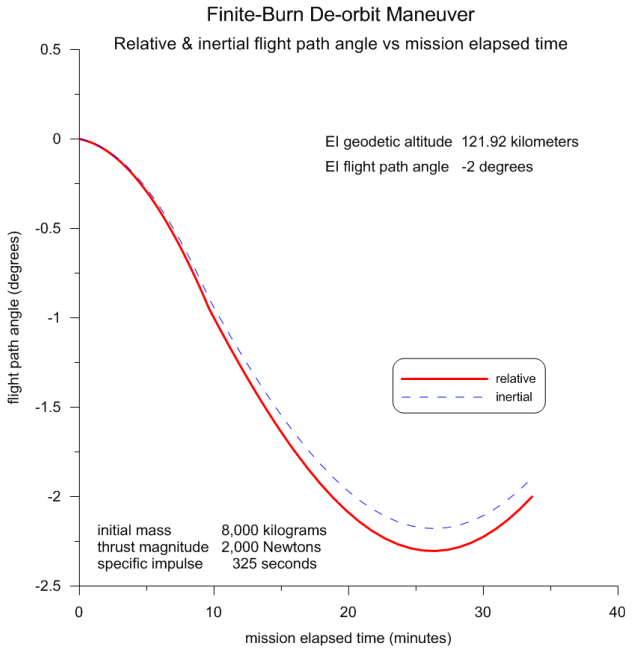
In additional to the user-defined solution output file, the `deorbit_sos` computer program will create a comma-separated-variable data file named `maneuver.csv`. This data file contains the information described later in this document (Contents of the simulation summary and csv files) starting at ignition and ending at termination of the propulsive maneuver.

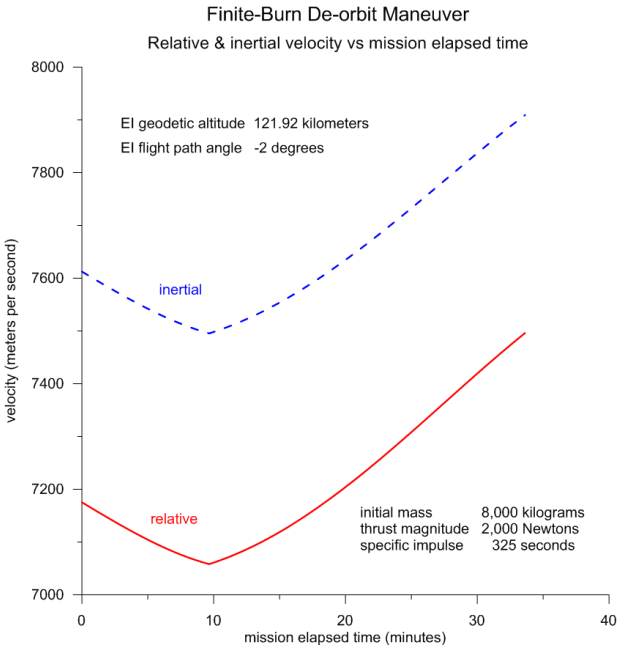
The following are graphic displays of several important flight conditions for this example. The first two images illustrate the behavior of the orbit-relative pitch and yaw angles, and the inertial right ascension and declination angles of the unit thrust vector during the de-orbit maneuver.



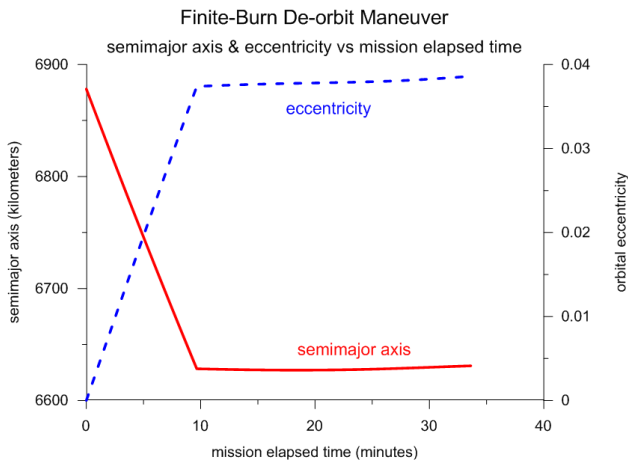


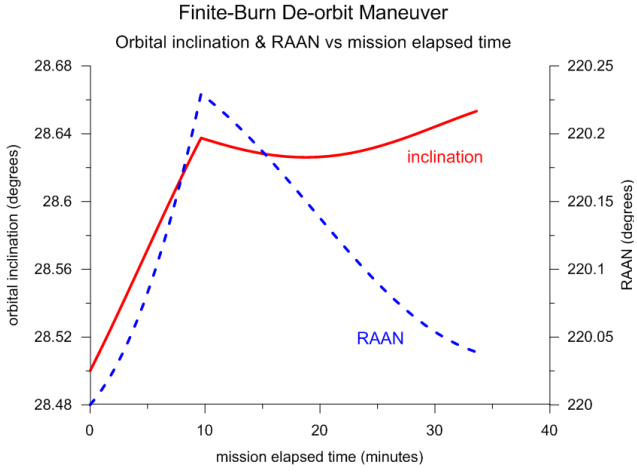
These two plots summarize the relative and inertial flight path angles and velocities.





The next two plots illustrate the behavior of the semimajor axis, orbital eccentricity, orbital inclination, and right ascension of the ascending node (RAAN) during the simulation.





Creating an initial guess

The software allows the user to input either a delta-v or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. For either type of initial guess, the user should also provide lower and upper bounds for the total thrust duration.

An estimate of the thrust duration can be determined from the following expression

$$t_d = \frac{I_{sp} m_p g}{F} = \frac{m_p V_{ex}}{F}$$

The propellant mass required for a given ΔV is a function of the initial (or final) mass of the spacecraft and the exhaust velocity as

$$m_p = m_i \left(1 - e^{\frac{-\Delta V}{V_{ex}}} \right) = m_f \left(e^{\frac{\Delta V}{V_{ex}}} - 1 \right)$$

In these equations

m_i = initial mass
m_f = final mass
m_p = propellant mass
V_{ex} = exhaust velocity = $g I_{sp}$
I_{sp} = specific impulse
ΔV = impulsive velocity increment
F = thrust
g = acceleration of gravity

The software uses a tangential thrusting steering method to generate an initial guess for the optimal trajectory. For tangential thrusting opposite to the flight path, the unit thrust vector in the modified equinoctial frame is simply $\mathbf{u}_T = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^T$. Please note that this type of steering creates a *coplanar* initial guess.

The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine for this aerospace trajectory optimization problem. These program options create an initial guess from the numerical integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* software to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 2` program option tells the program to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

An initial guess for the modified equinoctial orbital elements at the beginning of the final coast phase are determined by numerically integrating the equations of motion using the initial orbital elements, spacecraft mass, and propulsive characteristics provided by the user.

Algorithms for estimating the impulse de-orbit delta-v for both initial circular and elliptical orbits is explained in *Appendix E – Impulsive De-orbit from Earth Orbit*.

Problem setup

This part of the user's manual provides details about the software implementation within `deorbit_sos`. It defines such things as point and path constraints (boundary conditions), bounds on the dynamic variables, and the performance index or objective function.

Point functions – initial orbit constraints

The software allows the user to select one of the following initial orbit constraint options:

- constrain semimajor axis, eccentricity, and inclination
- constrain all initial orbital elements
- option 2 with unconstrained true longitude

For option 1, the initial orbit inclination is constrained by enforcing $\sqrt{h^2 + k^2} = \tan(i/2)$ where *i* is the initial orbit inclination provided by the user.

If the initial orbit is circular, the software enforces the following two equality constraints.

$$f = 0 \text{ and } g = 0$$

Otherwise, for an elliptical initial orbit, the single equality constraint $\sqrt{f^2 + g^2} = e$ is enforced, where *e* is the initial orbit eccentricity.

For program option 2, both lower and upper bounds for all modified equinoctial elements are set equal to the initial modified equinoctial orbital elements according to

$$p_L = p_U = p_i \qquad f_L = f_U = f_i$$

$$g_L = g_U = g_i \qquad h_L = h_U = h_i$$

$$k_L = k_U = k_i$$

Option 3 is identical to option 2 with the initial true longitude unbounded. In optimal control terminology, these derived constraints or boundary conditions are called *point functions*.

Performance index – maximize final spacecraft mass

The objective function or performance index *J* for this simulation is the mass of the spacecraft at burnout or termination of the propulsive maneuver. This is simply $J = m_f$. The value of the `maxmin` indicator in the *Sparse Optimization Suite* algorithm tells the software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the user-defined initial value.

Path constraint – unit thrust vector scalar magnitude

For a *variable steering* trajectory, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as $\left|\mathbf{u}_T\right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$.

Point functions – entry interface mission constraints

The entry interface mission constraints are relative flight path coordinates defined with respect to a rotating Earth. They are calculated from the inertial spacecraft coordinates at the entry interface using the algorithm described in *Appendix B – Trajectory Modeling in the Flight Path System.*

This set of possible equality constraints consists of the following elements

$$h_p - h_u = 0 \quad \rightarrow \text{geodetic altitude}$$

$$\gamma_p - \gamma_u = 0 \quad \rightarrow \text{relative flight path angle}$$

$$\phi_p - \phi_u = 0 \quad \rightarrow \text{geodetic latitude}$$

$$\lambda_p - \lambda_u = 0 \quad \rightarrow \text{east longitude}$$

$$\psi_p - \psi_u = 0 \quad \rightarrow \text{relative azimuth}$$

$$v_p - v_u = 0 \quad \rightarrow \text{relative velocity}$$

where the *p* subscript refers to values predicted by the software and the *u* subscript are the values defined by the user.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the orbital transfer.

$$0.05 m_{sc_i} \le m_{sc} \le 1.05 m_{sc_i} \qquad 100\, p_f \le p \le 0.8\, p_i$$

$$-1 \le f \le +1 \qquad -1 \le g \le +1$$

$$-1 \le h \le +1 \qquad -1 \le k \le +1$$

where m_{sc_i} is the initial spacecraft mass.

Finally, the three components of the unit thrust vector are constrained as follows

$$-1.1 \le u_r \le +1.1 \qquad -1.1 \le u_t \le +1.1 \qquad -1.1 \le u_n \le +1.1$$

Technical discussion

In this computer program, the orbital motion of the spacecraft is modeled in the modified equinoctial orbital elements coordinate system. Please refer to *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for a definition of these orbital elements along with the equations of motion. An algorithm for converting from geocentric declination and radius to geodetic altitude and latitude is described in *Appendix G – Aerospace Trajectory Coordinates and Time Systems.*

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)}\hat{\mathbf{u}}_T$$

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_r} & u_{T_t} & u_{T_n} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system. *The components of this unit vector are the control variables.*

The propellant mass flow rate is determined from

$$\dot{m} = \frac{dm}{dt} = \frac{T}{g\, I_{sp}}$$

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g\, I_{sp}$ is also called the *exhaust velocity*. The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}\, t$ where m_{sc_i} is the initial mass of the spacecraft and $\dot{m}$ is the propellant flow rate.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

Finally, the pitch and yaw angles can be determined from the components of the unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the "local horizontal" and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four-quadrant operation.

Example de-orbit from an elliptical earth orbit

This section illustrates the characteristics for a typical de-orbit from a highly elliptical Earth orbit (HEO). For this example, the entry interface constraints consist of the geodetic altitude and relative flight path angle. An estimate for the delta-v required for this example and the coast time were determined using the algorithm described in *Appendix E – Impulsive De-orbit from Earth Orbit.*

The main portion of the simulation definition file for this example is as follows:

```
*********************************************
** de-orbit trajectory optimization
** single finite-burn maneuver with final coast
** program deorbit_sos
** heo2ei.in - May 10, 2012
*********************************************
```

```
maneuver calendar date (month, day, year)
-----------------------------------------
3, 18, 2010

maneuver UTC (hour, minute, second)
-----------------------------------
12, 30, 45.875

initial spacecraft mass (kilograms)
8000.0

thrust magnitude (newtons)
1000.0

specific impulse (seconds)
325.0

**********************************
* type of propulsive initial guess *
**********************************
1 = thrust duration
2 = delta-v
-----------
2

initial guess for delta-v (meters/second)
35.0d0

initial guess for thrust duration (seconds)
700.0

lower bound for thrust duration (seconds)
1.0

upper bound for thrust duration (seconds)
1000.0

******************
* coast maneuver *
******************

initial guess for coast duration (minutes)
300.0

lower bound for coast duration (minutes; > 0)
200.0

upper bound for coast duration (minutes)
500.0

*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
24414.0d0

orbital eccentricity (non-dimensional)
0.727044

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
270.0

right ascension of the ascending node (degrees)
220.0d0

true anomaly (degrees)
180.0d0

**********************************
* initial orbit constraint options *
**********************************
```

```
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-------------------------------------------
3

*********************************************************
* entry interface constraints (set to 1.0d99 to ignore) *
*********************************************************

geodetic altitude (kilometers)
121.92d0

relative flight path angle (degrees)
-2.0d0

geodetic latitude (degrees)
1.0d99

east longitude (degrees)
1.0d99

relative azimuth (degrees)
1.0d99

relative velocity (meters/second)
1.0d99

************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate gravity model
------------------------
2
```

The program output for this example is

```
 program deorbit_sos
 ===================

 input data file ==> heo2ei.in

 oblate earth gravity model

 initial epoch
 -------------

 calendar date            March 18, 2010

 UTC time                 12:30:45.875

 ------------------------
 beginning of finite burn
 ------------------------

 mission elapsed time     00:00:00.000

      sma (km)              eccentricity       inclination (deg)       argper (deg)
  0.244140000000D+05    0.727044000000D+00    0.285000000000D+02    0.270000000000D+03

     raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
  0.220000000000D+03    0.179838597519D+03    0.898385975188D+02    0.632729583647D+03

      rx (km)                 ry (km)              rz (km)              rmag (km)
  0.237268343662D+05    -.284613210725D+05    0.201186544321D+05    0.421636066104D+05

      vx (kps)                vy (kps)             vz (kps)             vmag (kps)
  0.123989347177D+01    0.102137536270D+01    0.791045192336D-02    0.160642647766D+01

 ------------------
 end of finite burn
 ------------------
```

```
mission elapsed time     00:02:25.905

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.243269419485D+05    0.733224470367D+00    0.284999999981D+02    0.269999908966D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.220000048521D+03    0.180155324772D+03    0.901552337382D+02    0.629348220341D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.239053722555D+05    -.283115456389D+05    0.201186693081D+05    0.421636252457D+05

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 0.120740294448D+01    0.103164365370D+01    -.769905249676D-02    0.158813405416D+01

final mass               7954.22115070684      kilograms

propellant mass          45.7788492931622      kilograms

thrust duration          145.904574520815      seconds
                         2.43174290868026      minutes

delta-v                  18.2904541073263      meters/second

---------------------------
beginning of coast maneuver
---------------------------

mission elapsed time     00:02:25.905

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.243269419485D+05    0.733224470367D+00    0.284999999981D+02    0.269999908966D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.220000048521D+03    0.180155324772D+03    0.901552337382D+02    0.629348220341D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.239053722555D+05    -.283115456389D+05    0.201186693081D+05    0.421636252457D+05

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 0.120740294448D+01    0.103164365370D+01    -.769905249677D-02    0.158813405416D+01

---------------------
end of coast maneuver
---------------------

mission elapsed time     05:15:04.421

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.243574533818D+05    0.733691641752D+00    0.284878735015D+02    0.270125093234D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.219924531651D+03    0.355464345323D+03    0.265589438557D+03    0.630532606936D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.326983940138D+04    0.468558859467D+04    -.308885964145D+04    0.649520161919D+04

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 -.815487392014D+01    -.631067229242D+01    -.213830021530D+00    0.103136936504D+02

coast duration           18758.5159416022      seconds
                         312.641932360037      minutes
                         5.21069887266728      hours

relative flight path coordinates at entry interface
---------------------------------------------------

geodetic altitude     121.919999999998      kilometers

geodetic latitude    -28.5540897239011      degrees

east longitude        42.2691975996772      degrees

flight path angle    -2.00000000000468      degrees
```

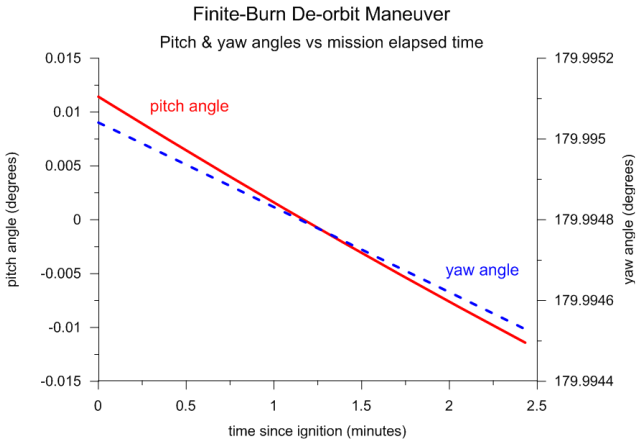
```
azimuth              92.4904131009538      degrees

velocity             9897.66361056145      meters/second

inertial flight path coordinates at entry interface
---------------------------------------------------

right ascension      124.909273063683      degrees

flight path angle   -1.91929389968213      degrees

azimuth              92.3897809434936      degrees

velocity             10313.6936503760      meters/second
```

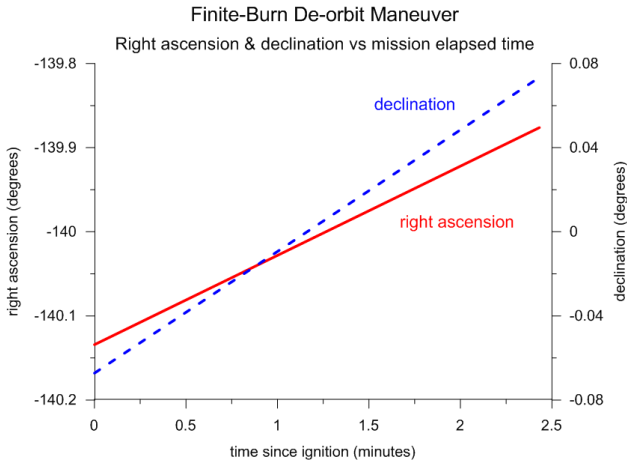
Here are the verification results for this example.

```
=======================================
verification of optimal control solution
=======================================

beginning of coast maneuver
---------------------------

mission elapsed time     00:02:25.905

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.243269419484D+05    0.733224470367D+00    0.284999999981D+02    0.269999908963D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.220000048522D+03    0.180155324775D+03    0.901552337371D+02    0.629348220339D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.239053722554D+05    -.283115456388D+05    0.201186693081D+05    0.421636252456D+05

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.120740294437D+01    0.103164365384D+01    -.769905257556D-02    0.158813405416D+01

end of coast maneuver
---------------------

mission elapsed time     05:15:04.421

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.243574533952D+05    0.733691641944D+00    0.284878734954D+02    0.270125093237D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.219924531681D+03    0.355464335541D+03    0.265589428778D+03    0.630532607457D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.326983852645D+04    0.468558927113D+04    -.308885961732D+04    0.649520165524D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.815487442942D+01    -.631067156466D+01    -.213830504622D+00    0.103136936178D+02

relative flight path coordinates at entry interface
---------------------------------------------------

geodetic altitude    121.920035928964      kilometers

geodetic latitude   -28.5540893076028      degrees

east longitude       42.2691865228324      degrees

flight path angle   -2.00000431339059      degrees

azimuth              92.4904186061209      degrees

velocity             9897.66357674878      meters/second

inertial flight path coordinates at entry interface
---------------------------------------------------

right ascension      124.909261986838      degrees
```

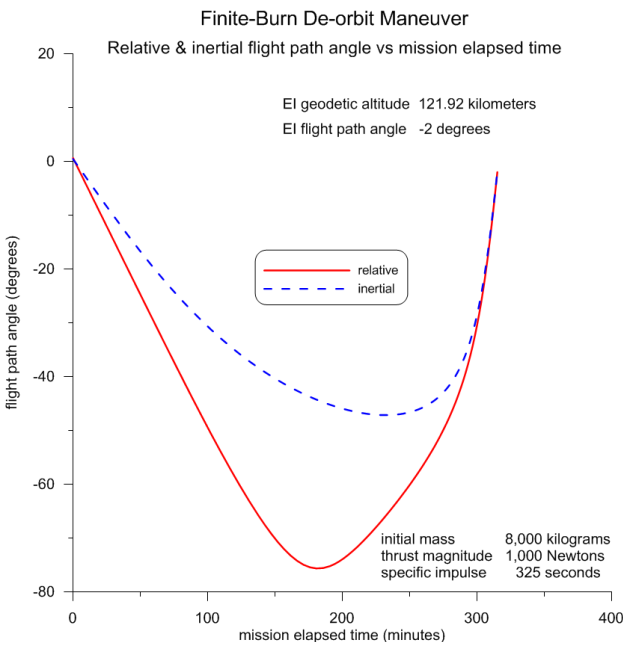
```
flight path angle    -1.91929803838563      degrees

azimuth               92.3897862248354      degrees

velocity              10313.6936177802      meters/second

mass and propulsive properties
------------------------------

final mass            7954.22115071184      kilograms

propellant mass       45.7788492881609      kilograms

thrust duration       145.904574520815      seconds
                      2.43174290868026      minutes

delta-v               18.2904541154479      meters/second
```

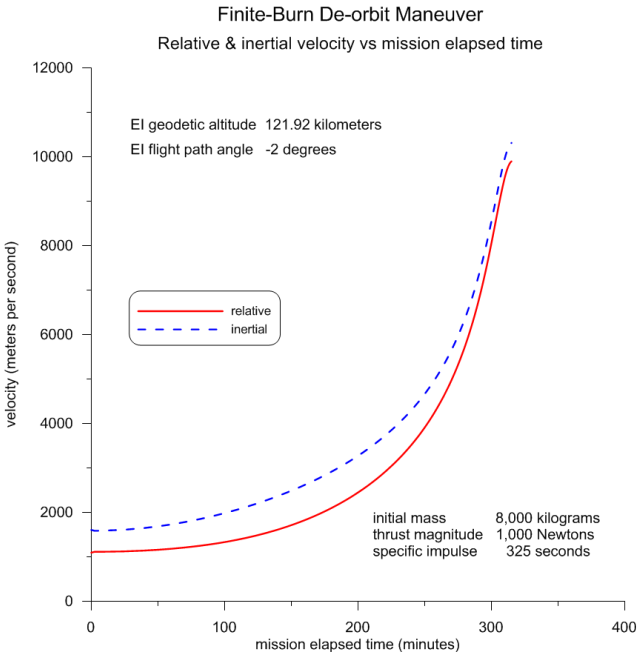
Here are plots of the control behavior and evolution of several important trajectory characteristics.





These two plots illustrate the behavior of the relative and inertial flight path angle and velocity.





Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `deorbit_sos` software.

The simulation summary screen display contains the following information:

mission elapsed time = simulation time (hh:mm:ss.sss)

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (min) = orbital period in minutes

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

geodetic altitude = geodetic altitude in kilometers

geodetic latitude = geodetic latitude in degrees)

east longitude = east longitude in degrees

flight path angle = relative or inertial flight path angle in degrees

azimuth = relative or inertial azimuth angle in degrees

velocity = relative or inertial velocity in meters per second

final mass = final spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The delta-v magnitude is determined using a cubic spline integration of the thrust acceleration data at each collocation node or user-defined step size.

The user-defined comma-separated-variable (csv) disk files is created by the `odeprt` subroutine and contains the following information:

time (sec) = mission elapsed time in seconds

time (min) = mission elapsed time in minutes

semimajor axis (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

arg of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (min) = orbital period in minutes

mass (kg) = spacecraft mass in kilograms

thracc (mps/s) = thrust acceleration in meters/second**2

perigee altitude = perigee altitude in kilometers

apogee altitude = apogee altitude in kilometers

geodetic altitude (km) = geodetic altitude in kilometers

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

ut-eci-x = x-component of eci unit thrust vector

ut-eci-y = y-component of eci unit thrust vector

ut-eci-z = z-component of eci unit thrust vector

semi-parameter = orbital semi-parameter in kilometers

f equinoctial element = modified equinoctial orbital element

g equinoctial element = modified equinoctial orbital element

h equinoctial element = modified equinoctial orbital element

k equinoctial element = modified equinoctial orbital element

true longitude = true longitude in degrees

rx (km) = x-component of the spacecraft’s position vector in kilometers

ry (km) = y-component of the spacecraft’s position vector in kilometers

rz (km) = z-component of the spacecraft’s position vector in kilometers

rmag (km) = magnitude of spacecraft's position vector in kilometers

vx (km) = x-component of the spacecraft's velocity vector in kilometers/second

vy (km) = y-component of the spacecraft's velocity vector in kilometers/second

vz (km) = z-component of the spacecraft's velocity vector in kilometers/second

vmag (km) = magnitude of spacecraft's velocity vector in kilometers/second

rasc (deg) = inertial right ascension of the unit thrust vector in degrees

decl (deg) = inertial declination of the unit thrust vector in degrees

yaw (deg) = out-of-plane yaw angle of the unit thrust vector in degrees

pitch (deg) = in-plane pitch angle of the unit thrust vector in degrees

declination (deg) = geocentric declination in degrees

geodetic lat (deg) = geodetic latitude in degrees

longitude (deg) = east longitude in degrees

azimuth (deg) = relative azimuth in degrees

vmagr (mps) = relative velocity in meters per second

fpar (deg) = relative flight path angle in degrees

fpai (deg) = inertial flight path angle in degrees

deltav (mps) = accumulative delta-v in meters per second

Hypersonic and Planetary Entry Flight Mechanics, Vinh, Busemann and Culp, The University of Michigan Press, 1980.

"Survey of Numerical Methods for Trajectory Optimization", John T. Betts, AIAA *Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998.

"Nearly Circular Transfer Trajectories for Descending Satellites", George M. Low, NASA Technical Report R-3, 1959.

"Optimum Deboost Altitude for Specified Atmospheric Entry Angle", Jerome M. Baker, Bruce E. Baxter, and Paul D. Arthur, *AIAA Journal*, Vol. 1, No. 7, July 1963.

"Deboost from Circular Orbits", A. H. Milstead, *The Journal of the Astronautical Sciences*, Vol. XIII, No. 4, pp. 170-171, July-August, 1966.

"Geometric Theory of Optimum Disorbit Problems", A. Busemann and N. X. Vinh, NASA CR-750, April 1967.

"Autonomous Optimal Deorbit Targeting", Donald J. Jezewski, AAS 91-136, AAS/AIAA Spaceflight Mechanics Meeting, February 11-13, 1991.

CMATO 8 – Lunar Ascent Trajectory Optimization

This *CMATO* application is a Fortran computer program named `lascent_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the single-stage, finite-burn lunar ascent trajectory optimization problem. The trajectory is modeled with a user-defined vertical rise phase, pitch-over phase, and initial and final boundary conditions for an ascent-to-orbit phase. This computer program attempts to maximize the final mass placed into lunar orbit or minimize the time to ascend to the final orbit. The lunar mission orbit can be circular or elliptical.

The important features of this scientific simulation are as follows:

- user-defined vertical rise and pitch-over phases
- angle-of-attack, bank angle and throttle control variables
- 3-DOF flight path equations of motion relative to a spherical, rotating Moon
- user-defined vehicle mass and propulsion properties
- user-defined mission orbit semimajor axis, eccentricity and inclination

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, "Practical Methods for Optimal Control and Estimation Using Nonlinear Programming" by John. T. Betts, SIAM, 2010.

The `lascent_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's problem and perform the optimization using a sparse nonlinear programming method.

The software allows the user to select the type of initial guess, collocation method and other important algorithm control parameters.

Input file format and contents

The `lascent_sos` software is "data-driven" by a user-created simulation definition text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in times font.

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly.

This data file defines a typical maximum payload simulation with throttling enabled and a circular final mission orbit with an inclination constraint. For this simulation, the initial azimuth is bounded so that the software can determine the correct azimuth after the pitch-over maneuver.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*********************************
** lunar launch vehicle simulation
** lv1.in
** November 15, 2005
** maximize payload to orbit
*********************************
```

The first input is an integer that defines the type of simulation. Option 2 is equivalent to a full throttle or constant thrust simulation.

```
type of trajectory optimization
*******************************
1 = maximize final mass
2 = minimize flight time
------------------------
1
```

The next three inputs allow the user to specify the duration of the vertical rise and pitch-over maneuvers along with the total pitch angle.

```
vertical rise time (seconds)
10

pitch over time duration (seconds)
30

pitch angle change during pitch over (degrees)
1.25
```

The next two inputs define the propulsion characteristics for the scientific simulation. Here the user can specify the maximum thrust and specific impulse.

```
*************************
propulsion characteristics
*************************

maximum thrust (newtons)
5000.0
```

```
specific impulse (seconds)
350.0
```

The initial vehicle mass and a guess for the final mass are defined by the next two data inputs.

```
****************************
vehicle mass characteristics
****************************

initial vehicle mass (kilograms)
3000.0

initial guess for final vehicle mass (kilograms)
1500.0
```

The following series of data items are reserved for user-defined initial conditions. To fix one or more initial conditions, the user should input identical lower and upper bounds. To free or un-constrain one or more initial states, set the lower and/or upper bounds to `1.0d99`. Please note the units and valid data range for each item.

```
***********************************
initial flight conditions and bounds
***********************************
NOTE 1: set upper and lower bounds to
the initial value to constrain or
"fix" the flight azimuth.
NOTE 2: set bound to 1.0d99 to ignore
-----------------------------------

initial flight azimuth (0 <= azimuth <= 360; degrees)
90.0

upper bound for initial flight azimuth (0 <= azimuth <= 360; degrees)
360.0

lower bound for initial flight azimuth (0 <= azimuth <= 360; degrees)
0.0
initial selenocentric latitude (-90 <= latitude <= + 90; degrees)
20.0

initial selenographic east longitude (0 <= longitude <= 360; degrees)
40.0
```

The following series of data items allow the user to define the final flight conditions. To fix one or more conditions, the user should input identical lower and upper bounds. To free or un-constrain one or more final states set the lower and/or upper bounds to `1.0d99`. Again, please note the units and valid data range for each item.

```
************************************************
initial guess for final flight conditions and bounds
************************************************
NOTE 1: set upper and lower bounds
to the final value to constrain or
"fix" a flight condition.
NOTE 2: set bound to 1.0d99 to ignore
-----------------------------------

final time (seconds)
800.0

upper bound for final time (seconds)
1.0d99
```

```
lower bound for final time (seconds)
1.0d99

final altitude (meters)
100000.0

upper bound for final altitude (meters)
120000.0

lower bound for final altitude (meters)
10000.0

final velocity (meters/second)
1600.0

upper bound for final velocity (meters/second)
2000.0

lower bound for final velocity (meters/second)
1000.0

final flight path angle (-90 <= fpa <= +90; degrees)
0.0

upper bound for final flight path angle (-90 <= fpa <= +90; degrees)
+90.0

lower bound for final flight path angle (-90 <= fpa <= +90; degrees)
-90.0

final flight azimuth (0 <= azimuth <= 360; degrees)
90.0

upper bound for final flight azimuth (0 <= azimuth <= 360; degrees)
360.0

lower bound for final flight azimuth (0 <= azimuth <= 360; degrees)
0.0

final declination (-90 <= declination <= +90; degrees)
0.0

upper bound for final declination (-90 <= declination <= +90; degrees)
+90.0

lower bound for final declination (-90 <= declination <= +90; degrees)
-90.0

final east longitude (0 <= longitude <= 360; degrees)
55.0

upper bound for final east longitude (0 <= longitude <= 360; degrees)
360.0

lower bound for final east longitude (0 <= longitude <= 360; degrees)
0.0
```

The next series of data inputs define lower and upper bounds on the state variables during the ascent-to-orbit flight phase. To free or un-constrain one or more states, set the lower and/or upper bounds to the value `1.0d99`.

```
************************************************************
upper and lower bounds on the flight conditions during ascent
************************************************************
NOTE: set bound to 1.0d99 to ignore
```

```
-----------------------------------

upper bound for altitude (meters)
120000.0

lower bound for altitude (meters)
1.0

upper bound for velocity (meters/second)
2000.0

lower bound for velocity (meters/second)
0.1

upper bound for flight path angle (-90 <= fpa <= +90; degrees)
+90.0

lower bound for flight path angle (-90 <= fpa <= +90; degrees)
-90.0

upper bound for flight azimuth (0 <= azimuth <= 360; degrees)
360.0

lower bound for flight azimuth (0 <= azimuth <= 360; degrees)
0.0

upper bound for declination (-90 <= declination <= +90; degrees)
+90.0

lower bound for declination (-90 <= declination <= +90; degrees)
-90.0

upper bound for east longitude (0 <= longitude <= 360; degrees)
360.0

lower bound for east longitude (0 <= longitude <= 360; degrees)
0.0
```

This next section of the input file defines the initial guesses and bounds for the control variables. To free or un-constrain one or more control variables set the lower and/or upper bounds to `1.0d99`.

```
*********************************
initial flight controls and bounds
*********************************
NOTE 1: set upper and lower bounds
to the initial value to constrain
or "fix" a flight control.
NOTE 2: set bound to 1.0d99 to ignore
-------------------------------------

initial angle-of-attack (degrees)
0.0

upper bound for initial angle-of-attack (degrees)
+90.0

lower bound for initial angle-of-attack (degrees)
-90.0

initial bank angle (degrees)
0.0

upper bound for initial bank angle (degrees)
+90.0
```

```
lower bound for initial bank angle (degrees)
-90.0

initial throttle setting
1.0

upper bound for initial throttle setting
1.0

lower bound for initial throttle setting
0.5
```

This section of the input file defines the final guesses and bounds for the control variables. To free one or more control variables set the lower and/or upper bounds to `1.0d99`.

```
*******************************
final flight controls and bounds
*******************************
NOTE 1: set upper and lower bounds
to the final value to constrain
or "fix" a flight control.
NOTE 2: set bound to 1.0d99 to ignore
-------------------------------------

final angle-of-attack (degrees)
10.0

upper bound for final angle-of-attack (degrees)
+90.0

lower bound for final angle-of-attack (degrees)
-90.0

final bank angle (degrees)
0.0

upper bound for final bank angle (degrees)
+90.0
lower bound for final bank angle (degrees)
-90.0

final throttle setting
0.75

upper bound for final throttle setting
1.0

lower bound for final throttle setting
0.5
```

This section of the input file defines the lower and upper bounds for the control variables during the ascent-to-orbit phase. To free or un-constrain one or more control variables, set the lower and/or upper bounds to `1.0d99`.

```
*********************************************************
upper and lower bounds on the flight controls during phase
*********************************************************
NOTE: set bound to 1.0d99 to ignore
-----------------------------------

upper bound for angle-of-attack (degrees)
+90.0

lower bound for angle-of-attack (degrees)
-90.0
```

```
upper bound for bank angle (degrees)
+90.0

lower bound for bank angle (degrees)
-90.0

upper bound for throttle setting
1.0

lower bound for throttle setting
0.5
```

These next three inputs allow the user to specify the final orbit characteristics. Please note that an input of `1.0d99` for the final orbital inclination will tell the *SOS* software to ignore the orbital inclination constraint.

```
**********************
final orbit constraints
**********************

final semimajor axis (kilometers)
1838.0

final orbital eccentricity (non-dimensional)
0.0d0

final orbital inclination (degrees; enter 1.0d99 to ignore this constraint)
25.0
```

The software also creates a comma-separated-variable (csv) ascii data file that contains the optimal control solution and other flight parameters. This next integer input specifies the type of solution data file to create.

```
****************************************
type of comma-delimited solution data file
****************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
----------------------------
1
```

For options 2 or 3, this next input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
1.0
```

The name of this output file is specified in the next line of information. Additional information about the contents of this file can be found in the Contents of the simulation summary csv file section later in this document.

```
name of comma-delimited solution data file
lv1.csv
```

The next series of program inputs are algorithm control options and parameters for the software. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
********************************
discretization/collocation method
********************************
```

```
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
------------------------------
1
```

The next integer defines the number of initial grid points to use in the collocation modeling of the ascent trajectory.

```
number of initial guess grid points to use
25
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*.

```
****************************
algorithm control parameters
****************************
```

This input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
100000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
**************************
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
```

```
**********************
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
------------------
1
```

The level of output from the differential equation algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
***************************
differential equation output
***************************
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
******************
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

Optimal control solution

Prior to performing the trajectory optimization the software will display the conditions at the beginning and end of the vertical rise, and the flight conditions at the end of the pitch-over maneuver. The following is the `lascent_sos` numerical output for this example.

```
 liftoff
 -------

 time          0.000000000000000E+000 seconds
 altitude      0.000000000000000E+000 meters
 longitude      40.0000000000000      degrees
 declination    20.0000000000000      degrees
 velocity      0.000000000000000E+000 meters/second
 fpa           0.000000000000000E+000 degrees
 azimuth       0.000000000000000E+000 degrees
 mass           3000.00000000000      kilograms

 end of vertical rise
 --------------------

 time           10.0000000000000      seconds
 altitude       2.31424039043486      meters
 longitude      40.0000000000000      degrees
 declination    20.0000000000000      degrees
 velocity      0.476409333678226      meters/second
 mass           2985.43262552889      kilograms
 delta-v        16.7072630675081      meters/second
```

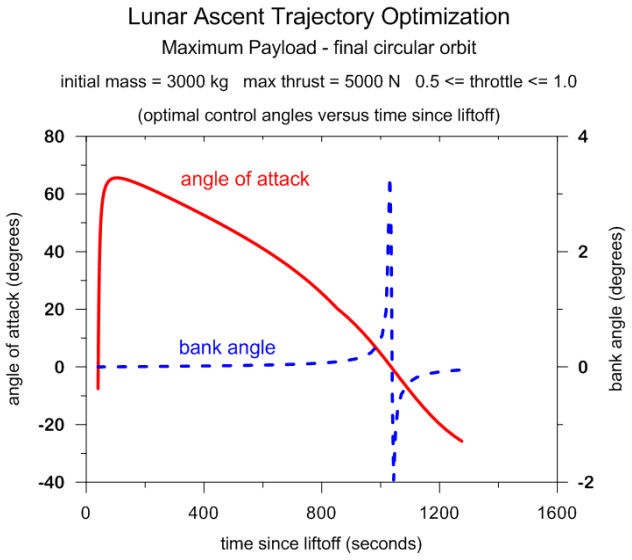
```
 end of pitch-over
 -----------------

 time           40.0000000000000      seconds
 altitude       43.5612765825354      meters
 longitude      40.0001936048219      degrees
 declination    20.0000000000000      degrees
 velocity       2.45921128624412      meters/second
 fpa            76.9988433933634      degrees
 azimuth        89.9999999999999      degrees
 aoa            11.7511558278279      degrees
 pitch          1.25000000000000      degrees
 mass           2941.73050211555      kilograms
 delta-v        50.6153500613182      meters/second
```

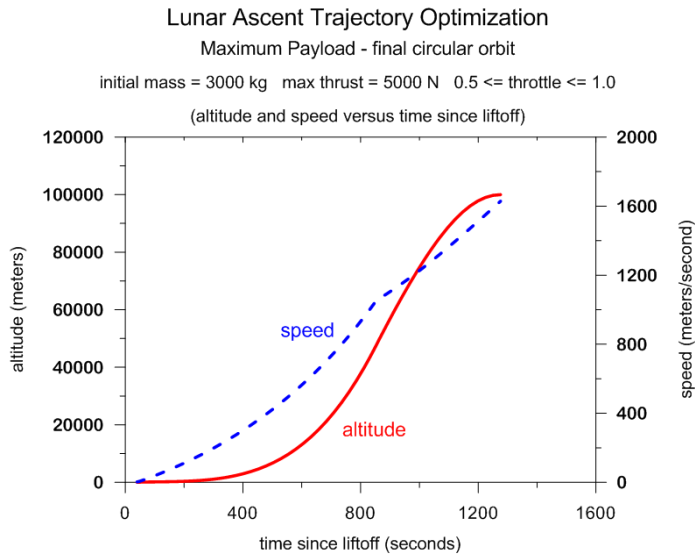
After convergence, the software will display the flight path conditions immediately after the pitch-over and the flight path conditions and classical orbital elements at mission orbit injection. The following is the program output for this example.

```
 program lascent_sos
 -------------------

 input data file ==> lascent1_max_payload.in

 maximize final mass

flight path coordinates after pitch over
 ----------------------------------------

 time                  40.0000000000000      seconds

 altitude              43.5612765827682      meters

 longitude             40.0001936048218      degrees

 declination           20.0000000000000      degrees

 relative velocity     2.45921128624421      meters/second

 relative fpa          76.9988433933634      degrees
 relative azimuth      105.173026738430      degrees

 vehicle mass          2941.73050211555      kilograms

final flight path coordinates
 -----------------------------

 time                  1275.75471116451      seconds

 altitude              100000.000000000      meters

 longitude             67.6748559760853      degrees

 declination           10.5792408056209      degrees

 relative velocity     1628.80391321129      meters/second

 relative fpa         1.785475748055718E-014 degrees

 relative azimuth      112.848287381972      degrees

 final mass and pitch-over to injection delta-v
 ----------------------------------------------

 vehicle mass          1448.95130106652      kilograms
```

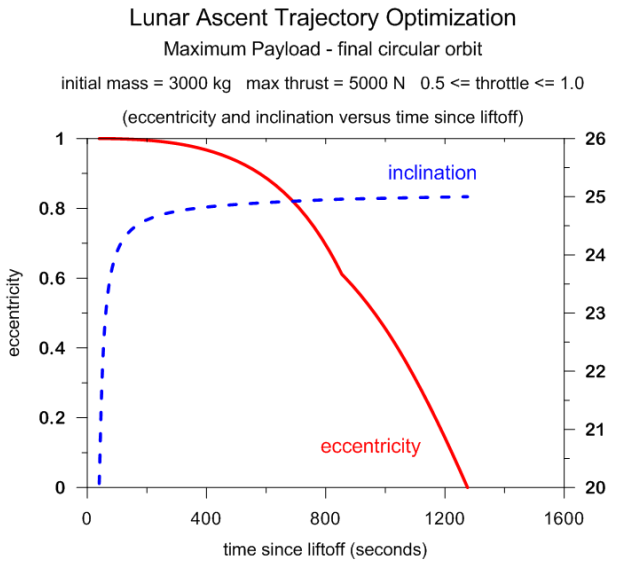
```
propellant mass        1551.04869893348      kilograms

delta-v                2430.21780258975      meters/second

mission orbit - orbital elements & state vector
-----------------------------------------------

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.183800000000D+04    0.123339131977D-14    0.250000000000D+02    0.000000000000D+00

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.271286134821D+03    0.154251398995D+03    0.154251398995D+03    0.117848695428D+03

     rx (km)                ry (km)               rz (km)               rmag (km)
 0.686318804299D+03    0.167132861969D+04    0.337447986892D+03    0.183800000000D+04

     vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 -.134883302933D+01    0.679412435911D+00    -.621701669512D+00    0.163323751027D+01
```

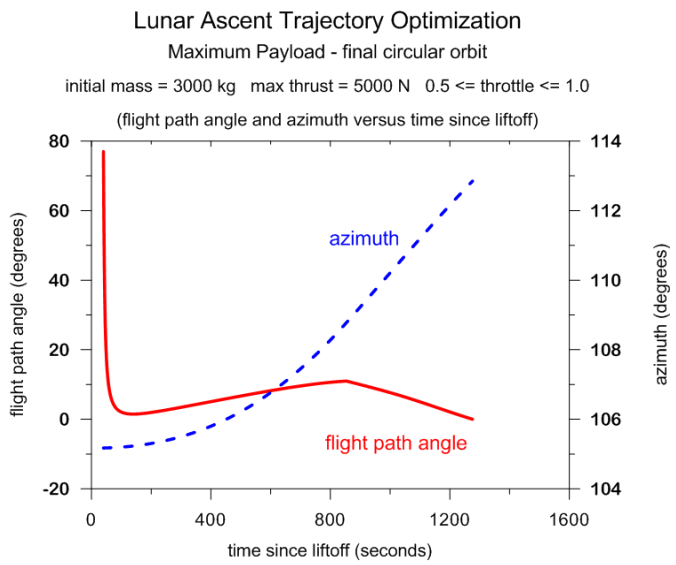
The following are trajectory plots created from the user-defined summary file. The first plot illustrates the behavior of the angle of attack and bank angle during the ascent trajectory. Notice the bank reversal as the angle of attack passes through zero. This behavior is explained on page 1192 of "Advanced Launch System Trajectory Optimization", by P. Daniel Burkhart and David G. Hull, AAS 96-168. An examination of the azimuthal differential equation given in the Technical discussion later in this document will help explain this behavior.



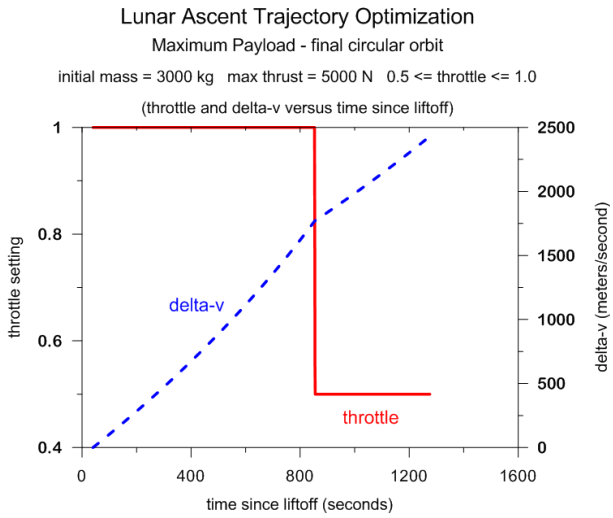


The next plot on the left illustrates the behavior of the orbital eccentricity and inclination during the ascent to the final circular mission orbit. The evolution of the relative flight path and azimuth angles is shown by the graphics on the right. These angles are computed with respect to a rotating Moon.





This final plot illustrates how the throttle setting and accumulated delta-v change during the ascent from the end of the pitch-over maneuver to injection into the final mission orbit.



Verification of the optimal control solution

The optimal solution determined by the *Sparse Optimization Suite* can be verified by numerically integrating the spacecraft's selenocentric equations of motion using the optimal control-computed initial conditions and the optimal control angles and throttle setting determined by the algorithm. This computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to explicitly integrate the flight path equations of motion.

The following is the program output summarizing the final flight path coordinates, classical orbital elements and state vector computed using this explicit numerical integration method. The vehicle mass and accumulated delta-v are computed by including the instantaneous propellant flow rate and thrust acceleration in the first-order equations of motion.

```
*****************************************
* verification of optimal control solution *
*****************************************
```

```
final flight path coordinates
-----------------------------

time                  1275.75471116451      seconds

altitude              99999.7802539489      meters

longitude             67.6748656532469      degrees

declination           10.5792368133830      degrees

relative velocity     1628.80441908298      meters/second

relative fpa        -1.066147781771430E-005 degrees

relative azimuth      112.848289156244      degrees

final mass and pitch-over to injection delta-v
----------------------------------------------

vehicle mass          1448.95131132294      kilograms

propellant mass       1551.04868867706      kilograms

delta-v               2430.63001693755      meters/second
```

Creating an initial guess

The software creates a linear interpolation guess based on the flight conditions after the pitch-over maneuver and the final flight conditions provided by the user. This is accomplished within the software by setting `init = 1`. The *Sparse Optimization Suite* attempts to obtain problem *feasibility* before trying to solve the optimal control problem.

If the software cannot find feasibility, the user will have to provide a better initial guess.

Problem setup

This section provides additional details about the *Sparse Optimization Suite* software implementation. It briefly explains such things as point constraints and the performance index options. The `lascent_sos` simulation consists of a propulsive vertical rise followed by a pitch-over maneuver and a final, optimized ascent-to-orbit propulsive phase.

Performance index

The performance index for the *maximize final mass* option is the final spacecraft mass given by $J = m_f = m\left(t_f\right)$ where m_f is the spacecraft mass at the final time.

For the *minimize flight time* program option, the performance index is $J = t_f$ where t_f is the simulation time at mission orbit injection.

The value of the `maxmin` indicator in the *Sparse Optimization Suite* algorithm tells the software whether the user is minimizing or maximizing the performance index.

Path constraints

The inequality path constraints enforced by the software are based on the lower and upper bounds for the dynamic variables provided by the user. For example,

$$\psi_L \le \psi \le \psi_U \qquad \gamma_L \le \gamma \le \gamma_U$$

and so forth. The subscripts *L* and *U* correspond to lower and upper bounds provided by the user.

Point constraints

The number and type of point functions enforced by the software is a function of the orbital elements of the final mission orbit. This section discusses the point constraints or final boundary conditions implemented in the `lascent_sos` computer program.

Circular orbit

For a final circular mission orbit, the user provides the semimajor axis of this orbit. The software then uses the following point constraints or boundary conditions to "target" a final circular orbit.

$$r_f = a_u \qquad v_f = \sqrt{\frac{\mu}{a_u}} \qquad \sin\gamma_f = \frac{\mathbf{r}_f \bullet \mathbf{v}_f}{\left|\mathbf{r}_f \bullet \mathbf{v}_f\right|} = 0$$

If the user also specifies an orbital inclination for the final mission orbit, the software enforces the following point constraint at the final time

$$\sin i_f = \frac{h_{f_z}}{h_f} = \frac{\left(\mathbf{r}_f \times \mathbf{v}_f\right)_z}{\left|\mathbf{r}_f \times \mathbf{v}_f\right|} = \sin i_u$$

In these equations, the *f* subscript refers to mission characteristics computed by the software at the final time and the *u* subscript refers to user-provide final or "target" orbital conditions.

Elliptical orbit

For a user-defined elliptical mission orbit, the software constrains the final specific orbital energy and final angular momentum magnitude according to

$$\mathrm{E}_f = v_f^2 - \frac{2\mu}{r_f} = \mathrm{E}_u = v_u^2 - \frac{2\mu}{r_u} \qquad h_f = \frac{\mathbf{r}_f \times \mathbf{v}_f}{\left|\mathbf{r}_f \times \mathbf{v}_f\right|} = h_u = \frac{\mathbf{r}_u \times \mathbf{v}_u}{\left|\mathbf{r}_u \times \mathbf{v}_u\right|}$$

If the user also specifies an orbital inclination for the final mission orbit, the software enforces the following point constraint

$$\sin i_f = \frac{h_{f_z}}{h_f} = \frac{\left(\mathbf{r}_f \times \mathbf{v}_f\right)_z}{\left|\mathbf{r}_f \times \mathbf{v}_f\right|} = \sin i_u$$

In these equations, the *f* subscript refers to mission characteristics computed by the software at the final time and the *u* subscript refers to user-provided final orbital conditions. The user-defined position vector $\mathbf{r}_u$ and velocity vector $\mathbf{v}_u$ are determined from the semimajor axis, orbital eccentricity and inclination input by the user. All other angular orbital elements are set to 0 degrees.

Technical discussion

The first-order flight path equations of motion relative to a rotating, spherical Moon are

selenocentric radius

$$\dot{r} = \frac{dr}{dt} = V \sin\gamma$$

selenographic longitude

$$\dot{\lambda} = \frac{d\lambda}{dt} = V \frac{\cos\gamma \sin\psi}{r \cos\delta}$$

selenocentric declination

$$\dot{\delta} = \frac{d\delta}{dt} = V \frac{\cos\gamma \cos\psi}{r}$$

speed

$$\dot{V} = \frac{dV}{dt} = \frac{T\cos\alpha}{m} - g\sin\gamma + \omega^2 r \cos\delta\left(\sin\gamma\cos\delta - \sin\delta\cos\gamma\cos\psi\right)$$

flight path angle

$$\dot{\gamma} = \frac{d\gamma}{dt} = \frac{V}{r}\cos\gamma + \left(\frac{T\sin\alpha}{mV}\right)\cos\beta - \frac{g\cos\gamma}{V} + 2\omega\sin\psi\cos\delta$$
$$+\omega^2 \frac{r}{V}\cos\delta\left(\cos\psi\sin\gamma\sin\delta + \cos\gamma\cos\delta\right)$$

flight azimuth

$$\dot{\psi} = \frac{d\psi}{dt} = \frac{V}{r}\tan\delta\sin\psi\cos\gamma + \left(\frac{T\sin\alpha}{mV\cos\gamma}\right)\sin\beta$$
$$+2\omega\left(\sin\delta - \cos\psi\cos\delta\tan\gamma\right) + \frac{r}{V\cos\gamma}\omega^2\sin\psi\cos\delta\sin\delta$$

where

$r =$ selenocentric radius	$V =$ relative speed
$\gamma =$ flight path angle	$\delta =$ selenocentric declination
$\lambda =$ selenographic longitude $(+$ east$)$	$\psi =$ flight azimuth $(+$ clockwise from north$)$
$\beta =$ bank angle $(+$ for a right turn$)$	$\alpha =$ angle of attack
$r_{eq} =$ lunar equatorial radius	$\omega =$ lunar inertial rotation rate
$\mu =$ lunar gravitational constant	$g =$ acceleration of gravity $= \mu/r^2$
$T =$ propulsive thrust	$m =$ spacecraft mass

Please refer to *Appendix B – Trajectory Modeling in the Flight Path System* for additional properties and relationships of the flight path coordinate system.

Vertical rise phase

During the vertical rise part of the trajectory, the software uses a subset of the full equations of motion. This set of equations includes $\dot{r}, \dot{v}, \dot{\delta}, \dot{\lambda}$ with $\dot{\gamma} = \dot{\psi} = 0$. This is necessary to avoid singularities in the flight path and azimuth differential equations during the trajectory evolution. Furthermore, during the numerical integration of the vertical rise differential equations, $\alpha = 0^\circ$, $\beta = 0^\circ$, $\gamma = 90^\circ$, $\psi = 0^\circ$ and the throttle setting is equal to `1`.

Pitch-over phase

During the pitch-over part of the trajectory, the software also uses a subset of the full equations of motion. This set of equations includes $\dot{r}, \dot{v}, \dot{\delta}, \dot{\lambda}, \dot{\gamma}$ with $\psi = 0^\circ$. This is necessary since azimuth is poorly defined until the end of the pitch-over part of the ascent trajectory. Furthermore, during the numerical integration of the pitch-over differential equations, $\alpha = 0^\circ, \beta = 0^\circ, \gamma = 90^\circ, \psi = 0^\circ$ and the throttle setting is equal to `1`.

The angle of attack during the pitch-over phase is determined from $\alpha = 90^\circ - \gamma + \dot{\theta}\left(t - t_p\right)$ where $\dot{\theta}$ is the pitch rate, *t* is the time since launch and t_p is the time since launch at which the pitch-over maneuver begins. The constant pitch rate is determined from the pitch angle and pitch-over maneuver duration provided by the user.

For a realistic and contiguous ascent trajectory, the launch vehicle should pitch over to the azimuth direction determined by the ascent-to-final orbit optimization.

Cross-range and downrange calculations

The cross-range angle is determined from the following expression which can be derived from spherical trigonometry relationships.

$$\sin\nu = -\sin\psi_1 \sin\phi_2 \cos\phi_1 \cos\Delta\lambda - \cos\psi_1 \cos\phi_1 \sin\Delta\lambda + \sin\psi_1 \cos\phi_2 \sin\phi_1$$

The downrange angle is determined from the following three equations

$$\sin\mu = -\cos\psi_1 \sin\phi_2 \cos\phi_1 \cos\Delta\lambda + \sin\psi_1 \cos\phi_1 \sin\Delta\lambda + \cos\psi_1 \cos\phi_2 \sin\phi_1$$

$$\cos\mu = \cos\phi_2 \cos\phi_1 \cos\Delta\lambda + \sin\phi_2 \sin\phi_1$$

$$\mu = \tan^{-1}\left(\sin\mu, \cos\mu\right)$$

where

$\phi_1 =$ selenocentric declination of the initial point

$\psi_1 =$ flight azimuth at the initial point

$\phi_2 =$ selenocentric declination of the final point

$\Delta\lambda = \lambda_2 - \lambda_1$

$\lambda_1 =$ east longitude of the initial point

$\lambda_2 =$ east longitude of the final point

Please note the inverse tangent above is a four-quadrant calculation.

Conversion of flight path coordinates to inertial state vector

The software uses the following expressions to calculate the selenocentric, inertial position and velocity vectors of the spacecraft at the final time.

$$r_x = r\cos\delta\cos\lambda \qquad r_y = r\cos\delta\sin\lambda \qquad r_z = r\sin\delta$$

$$v_x = v\left[\cos\lambda\left(-\cos\psi\sin\beta\sin\delta + \cos\beta\cos\delta\right) - \sin\psi\sin\beta\sin\alpha\right]$$

$$v_y = v\left[\sin\lambda\left(-\cos\psi\sin\beta\sin\delta + \cos\beta\cos\delta\right) + \sin\psi\sin\beta\cos\alpha\right]$$

$$v_z = v\left(\cos\psi\cos\delta\sin\beta + \cos\beta\cos\delta\right)$$

where the correction for lunar rotation is

$$v_x = v_x - \omega r_y \qquad v_y = v_y + \omega r_x$$

The inertial speed can also be computed from the following expression

$$v_i = \sqrt{v^2 + 2vr\omega\cos\gamma\sin\psi\cos\delta + r^2\omega^2\cos^2\delta}$$

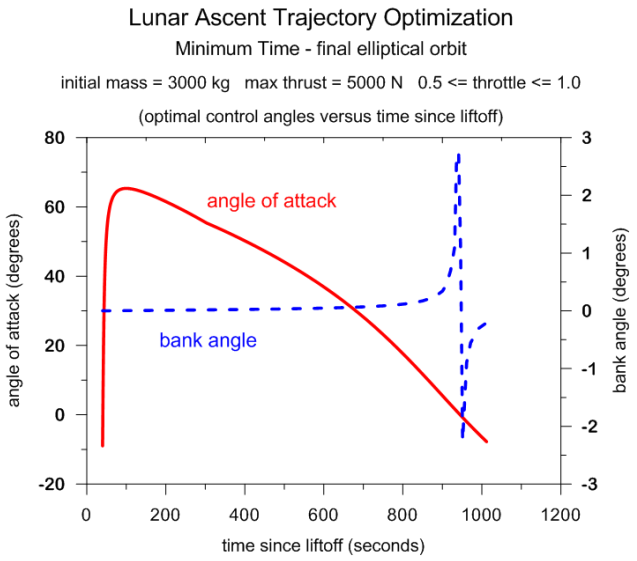
and the inertial flight path angle from

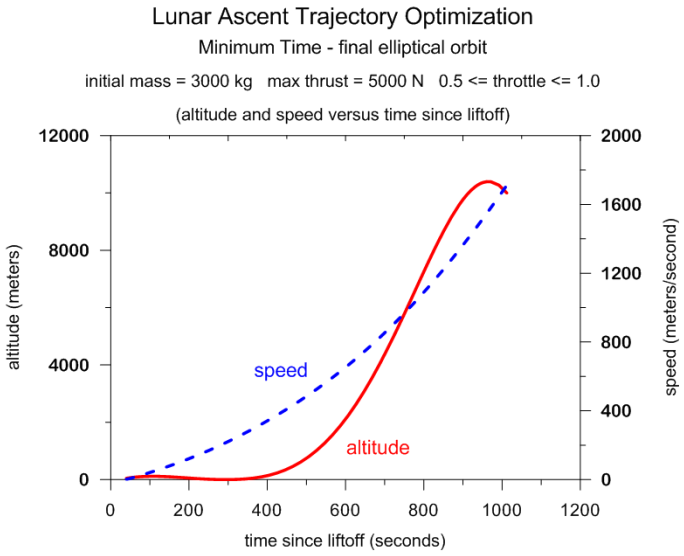
$$\cos\gamma_i = \sqrt{\frac{v^2\cos^2\gamma + 2vr\omega\cos\gamma\cos\psi\cos\delta + r^2\omega^2\cos^2\delta}{v^2 + 2vr\omega\cos\gamma\cos\psi\cos\delta + r^2\omega^2\cos^2\delta}}$$

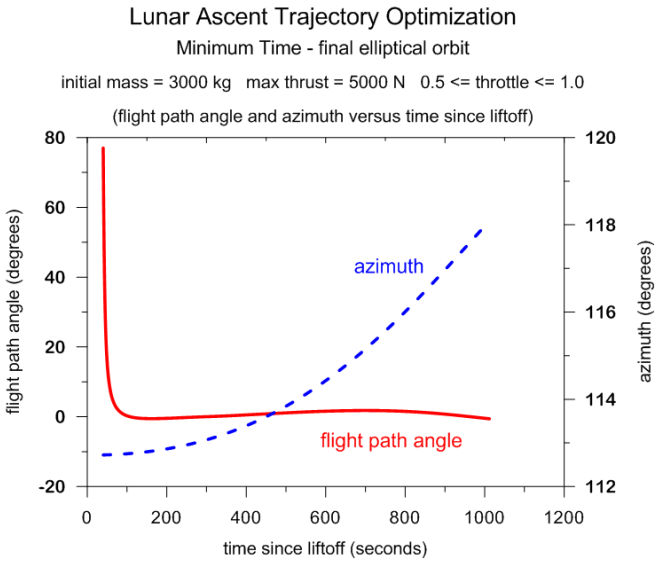
where all coordinates on the right-hand-side of these equations are relative to a rotating Moon.

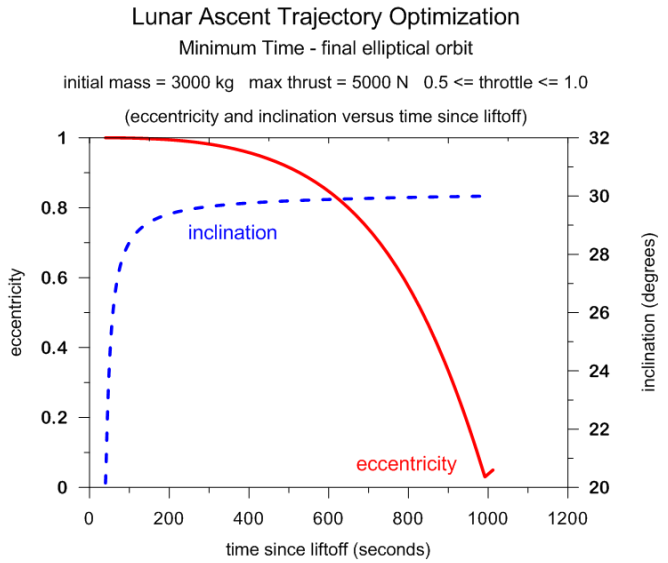
Minimum time, elliptical final orbit example

This section summarizes the `lascent_sos` solution and trajectory graphics for a *minimum time* ascent to an *elliptical* final mission orbit.









Since this is a minimum time solution, the throttle setting during the pitch-over to final mission injection is always 100 percent.

Here is the `lascent_sos` program output for this example.

```
program lascent_sos
-------------------

input data file ==> lascent1_min_time_elliptical.in

minimize total time

flight path coordinates after pitch over
----------------------------------------

time                  40.0000000000000      seconds

altitude              43.5612765830010      meters

longitude             40.0001936048218      degrees

declination           20.0000000000000      degrees
```

```
relative velocity     2.45921128624421      meters/second

relative fpa          76.9988433933635      degrees

relative azimuth      112.723576550163      degrees

vehicle mass          2941.73050211555      kilograms

final flight path coordinates
-----------------------------

time                  1012.03207649040      seconds

altitude              10000.0000000000      meters

longitude             59.0163724992820      degrees

declination           11.1909848017253      degrees

relative velocity     1711.24012729020      meters/second

relative fpa        -0.581594381564705      degrees

relative azimuth      118.088411735836      degrees

final mass and pitch-over to injection delta-v
----------------------------------------------

vehicle mass          1525.73497649851      kilograms

propellant mass       1474.26502350149      kilograms

delta-v               2253.39867838197      meters/second

mission orbit - orbital elements & state vector
-----------------------------------------------

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.183800000000D+04    0.500000000000D-01    0.300000000000D+02    0.169425377108D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.259056128591D+03    0.347734565247D+03    0.157159942355D+03    0.117848695428D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.882748192401D+03    0.147009109148D+04    0.339251840931D+03    0.174800000000D+04

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 -.122640620953D+01    0.898934666349D+00    -.793719576539D+00    0.171526863533D+01

*******************************************
* verification of optimal control solution *
*******************************************

final flight path coordinates
-----------------------------

time                  1012.03207649040      seconds

altitude              9997.50294686644      meters

longitude             59.0165121920944      degrees

declination           11.1909123546710      degrees

relative velocity     1711.24767834024      meters/second

relative fpa        -0.581726988142458      degrees

relative azimuth      118.088437416777      degrees

final mass and pitch-over to injection delta-v
----------------------------------------------

vehicle mass          1525.73497649861      kilograms
```

```
propellant mass       1474.26502350139      kilograms

delta-v               2253.39772068493      meters/second

mission orbit - orbital elements & state vector
-----------------------------------------------

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.183801231387D+04    0.500080360095D-01    0.299999991538D+02    0.169426460294D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.259056130199D+03    0.347733635676D+03    0.157160095970D+03    0.117849879739D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.882743567977D+03    0.147009151140D+04    0.339249188092D+03    0.174799750295D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.122641590194D+01    0.898931343902D+00    -.793724668909D+00    0.171527618057D+01
```

Contents of the simulation summary csv file

This section is a summary of the information contained in the CSV data file produced by the `lascent_sos` software. The comma-separated-variable disk file is created by the `odeprt` subroutine and contains the following information:

time (seconds) = simulation time since launch in seconds

altitude (meters) = altitude relative to a spherical Moon in meters

velocity (mps) = Moon-relative velocity in meters per second

flight path angle (d) = Moon-relative flight path angle in degrees

azimuth (deg) = Moon-relative azimuth angle in degrees

declination (deg) = selenocentric declination in degrees

longitude (deg) = selenographic east longitude in degrees

angle of attack (deg) = angle-of-attack in degrees

bank angle (deg) = bank angle in degrees

throttle = throttle setting

thrust (newtons) = thrust in newtons

load factor (g) = load factor in Earth g's

crossrange (km) = cross-range distance in kilometers

downrange (km) = downrange distance in kilometers

vehicle mass (kg) = vehicle mass in kilograms

fuel flowrate (kg/s) = fuel flowrate in kilograms/second

rmag (meters) = selenocentric distance in meters

vmag (mps) = selenocentric speed in meters/second

semimajor axis (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

arg of periapsis (deg) = argument of periapsis in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

periapsis radius (km) = selenocentric periapsis radius in kilometers

apoapsis radius (km) = selenocentric apoapsis radius in kilometers

eci fpa (deg) = inertial flight path angle in degrees

"Optimization of Thrust Direction Histories and Vehicle Parameters for Three-dimensional Ascent Trajectories", C. Tucker Battle and Robert G. Gottlieb, AIAA 64-663.

"Design and Evaluation of a 3-D Optimal Ascent Guidance Algorithm", Anthony J. Calise, Nahum Melamed and Seungjae Lee, AIAA 97-3707.

"Optimization of Launch Vehicle Ascent Trajectories with Path Constraints and Coast Arcs", C. P. Gath and A. J. Calise, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 24, No. 2, March-April, 2001, pp. 296-304.

"Advanced Launch System Trajectory Optimization Using Suboptimal Control", Douglas A. Shaver and David G. Hull, AIAA 90-3413.

"Solving the Optimal Control Problem Using a Nonlinear Programming Technique, Part 2: Optimal Shuttle Ascent Trajectories", T. Bauer, J. Betts, W. Huffman and K. Zondervan, AIAA 84-2038.

"Semi-Analytical Formulas for Launcher Performance Evaluation", Emanuele Di Sotto and Paolo Teofilatto, *AIAA Journal of Guidance, Control and Dynamics*, Vol. 25, No. 3, May-June, 2002, pp. 538-545.

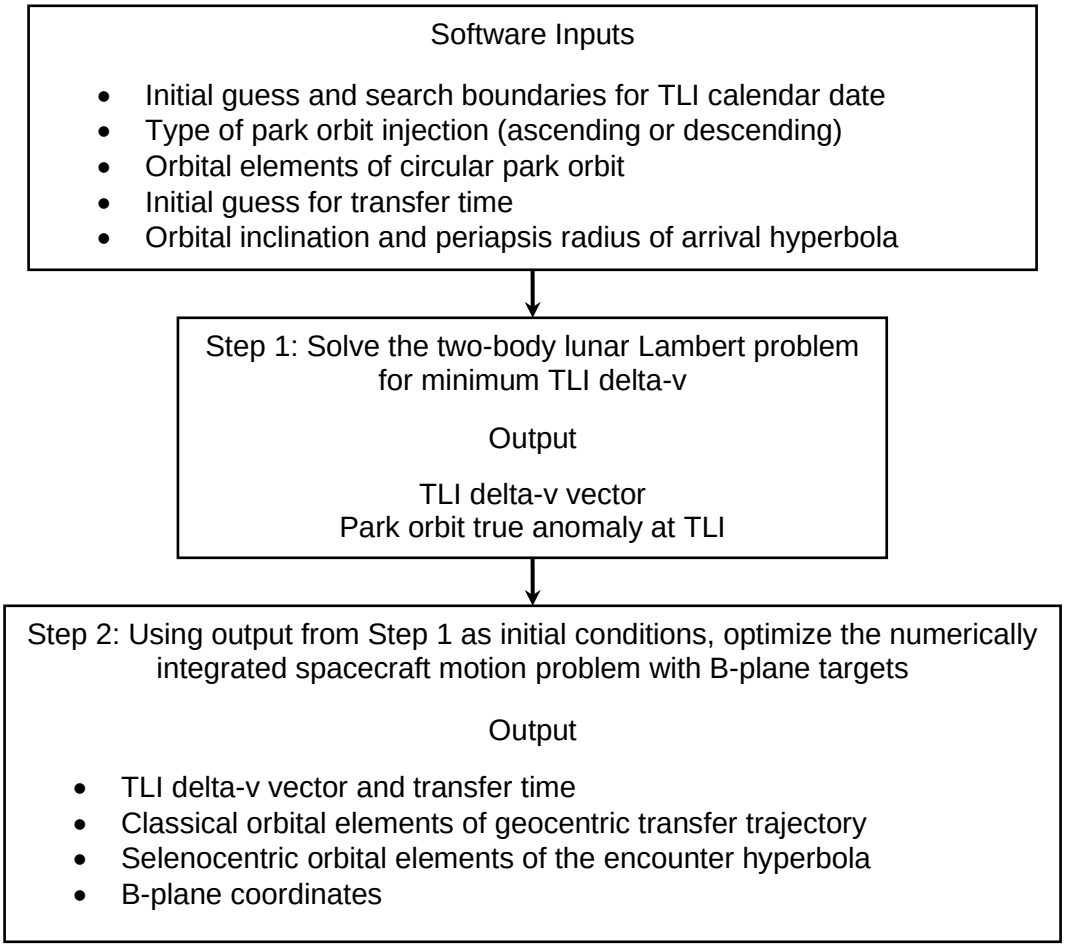
CAMTO 9 – Impulsive Trans-Lunar Trajectory Optimization

This *CMATO* application is a Fortran computer program named `tlto.exe` that can be used to design and optimize lunar missions from Earth park orbit to B-plane encounter at the moon. The software assumes that trans-lunar injection (TLI) occurs *impulsively* from a circular Earth park orbit. The B-plane coordinates and final orbital elements are expressed in a moon-centered (selenocentric) mean lunar equator and IAU node of epoch coordinate system.

The first part of the software solves for the minimum TLI impulsive delta-v using a two-body Lambert solution for the transfer trajectory from the Earth park orbit to the center of the moon. The second part of the computer program implements a simple *shooting* method that attempts to minimize the impulsive TLI delta-v while numerically integrating the spacecraft equations of motion and targeting to components of the B-plane.

In the shooting algorithm, the spacecraft motion model includes the Earth's non-spherical gravity effect and the point-mass perturbations of the sun and moon. The B-plane targets are enforced using the combination of user-defined periapsis radius and orbital inclination of the arrival hyperbola at the moon.

The program inputs and major computational steps implemented in this software are as follows



The lunar coordinates required by the software are computed using the JPL DE421 ephemeris.

Input data file format and contents

The `tlto` computer program is "data-driven" by a simple text file created by the user. This section describes a typical input data file. In the following discussion the actual input file contents are in *`courier`* font and all explanations are in times font. Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to

reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input.

The first five lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with five and only five initial text lines.

```
************************************
* input file for program tlto
* tlto1.in – December 4, 2010
* polar lunar orbit at 100 km altitude
************************************
```

The software allows the user to specify an initial guess for the TLI calendar date and lower and upper bounds on the actual date found during both the two-body and numerically integrated TLI delta-v optimization processes. For any guess for the TLI time t_{TLI} and user-defined lower and upper bounds Δt_l and Δt_u, the actual TLI time t is constrained as follows:

$$t_{TLI} - \Delta t_l \le t \le t_{TLI} + \Delta t_u$$

The first five inputs define the initial guesses for the TLI calendar date and the lower and upper bounds, respectively. Be sure to include all four digits of the calendar year. The first set of bounds is used during the two-body optimization, and the second set is used during the numerical integration and B-plane targeting.

The TLI calendar date is a control variable in the NLP formulation and must always have a lower and upper bound. For a fixed TLI calendar date, input small values (e.g., plus and minus `1.0e-8`) for the bounds.

```
initial guess for TLI calendar date (month, day, year)
9,15,2008

lower bound for TLI calendar date search (two-body optimization; hours)
0.0

upper bound for TLI calendar date search (two-body optimization; hours)
+24.0

lower bound for TLI calendar date search (integrated optimization; hours)
-12.0

upper bound for TLI calendar date search (integrated optimization; hours)
+12.0
```

The next input is the user's initial guess for the TLI-to-B-plane transfer time, in hours.

```
initial guess for transfer time (hours)
110.0
```

The next two numbers define the fixed values for park orbit altitude and orbital inclination. The altitude is measured with respect to a spherical Earth model.

```
*********************************
circular park orbit characteristics
*********************************

altitude (kilometers)
185.32

orbital inclination (degrees)
28.5
```

This next integer input defines the type of TLI maneuver to perform. The software uses this indicator to compute the park orbit RAAN.

```
type of TLI maneuver
(1 = ascending, 2 = descending)
```

```
2
```

This next integer input defines the type of targeting algorithm to use.

```
*****************************
type of final orbit targeting
-----------------------------
1 = periapsis radius and inclination
2 = user-defined b-plane coordinates
***********************************
1
```

The next two inputs define the periapsis radius and orbital inclination to use during the numerically integrated solution. These coordinates refer to the selenocentric hyperbola. The orbital inclination should be specified in the mean lunar equator and IAU node of epoch coordinate system.

```
----------------------------------------
final lunar orbit characteristics
(mean lunar equator and IAU node of epoch)
----------------------------------------

periapsis radius (kilometers)
1838.0

orbital inclination (degrees)
90.0
```

For targeting option 2, the software allows the user to input B-plane targets directly using the following two program inputs.

```
----------------------------------------
user-defined b-plane targets
(mean lunar equator and IAU node of epoch)
----------------------------------------

user-defined b dot r target (kilometers)
6000.0d0

user-defined b dot t target (kilometers)
-10.0d0
```

The next series of inputs define the types of perturbations to include during the numerical integration of the spacecraft's equations of motion. The first text input is the name of the Earth gravity model data file to use. Items 2 and 3 allow the user to specify the order and degree of the Earth gravity model in the equations of motion, and items 4 and 5 specify options to include the point mass gravity of the moon and sun.

```
***********************
trajectory perturbations
***********************

name of Earth gravity model data file
egm96.dat

order of Earth gravity model (zonals)
8

degree of Earth gravity model (tesserals)
8

include lunar perturbation (1 = yes, 0 = no)
1

include solar perturbation (1 = yes, 0 = no)
1
```

The next two inputs are algorithm control parameters. The first input is the truncation error tolerance for the Runge-Kutta-Felhberg integrator and determines how well the equations of orbital motion are solved. The second input is the root-finding tolerance and it determines how accurately close approach to the moon is predicted.

```
*****************************
algorithm control parameters
*****************************
truncation error tolerance
1.0d-12

root-finding tolerance
1.0d-8
```

The final two inputs specify the name of the solution disk file and the time step at which the data is created and written to this file.

```
**************************
output file characteristics
**************************

name of output data file
tlto1.csv

print step size (minutes)
10.0
```

Program example

The following is the solution created by the computer program for this example. The output is organized by the following major sections

- First pass
    - optimized two body Lambert solution
    - TLI delta-v vector and magnitude
    - pre-TLI and post-TLI flight conditions
- Targeting pass
    - pre-TLI and post-TLI flight conditions
    - TLI delta-v vector and magnitude
    - time and conditions at lunar closest approach
    - classical orbital elements of the lunar transfer trajectory
    - B-plane coordinates at closest approach to the moon

The first output section summarizes the optimized two-body Lambert solution. The solution is presented in the Earth mean equator of J2000 coordinate system (EME2000). The trajectory characteristics are given before and after the impulsive TLI maneuver. The event time is provided in both Universal Coordinated Time (UTC) and Barycentric Dynamical Time (TDB).

```
====================================
two-body lunar trajectory optimization
====================================

input data file ==> tlto1.in

descending transfer

time and conditions prior to TLI
(geocentric Earth mean equator and equinox J2000)
```

```
-------------------------------------------------

UTC calendar date       September 15, 2008

UTC time                13:29:13.655

UTC Julian date         2454725.06196359

TDB time                13:30:18.837

TDB Julian date         2454725.06271802

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.656345630000D+04    0.111646925668D-15    0.285000000000D+02    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.357108632971D+03    0.242927962813D+03    0.242927962813D+03    0.146996629514D+01

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.324237192596D+04    -.497888348087D+04    -.278867391059D+04    0.656345630000D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.677307078339D+01    -.346292269424D+01    -.169231904268D+01    0.779296254099D+01

time and conditions after TLI
(geocentric Earth mean equator and equinox J2000)
-------------------------------------------------

UTC calendar date       September 15, 2008

UTC time                13:29:13.655

UTC Julian date         2454725.06196359

TDB time                13:30:18.837

TDB Julian date         2454725.06271802

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.187780715202D+06    0.965047229195D+00    0.285000000000D+02    0.242927927168D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.357108632971D+03    0.356447377822D-04    0.242927962813D+03    0.224949464232D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.324237192596D+04    -.497888348087D+04    -.278867391059D+04    0.656345630000D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.949449861264D+01    -.485433249196D+01    -.237229666009D+01    0.109241859784D+02

trans-lunar injection delta-v vector and magnitude
(geocentric Earth mean equator and equinox J2000)
-------------------------------------------------

deltav-x             2721.42782925067       meters/second
deltav-y            -1391.40979772003       meters/second
deltav-z            -679.977617407291       meters/second

deltav               3131.22343744202       meters/second

eci unit thrust vector

 0.869126040865955      -0.444366180031124      -0.217160362712021

rtn unit thrust vector

 1.065917010975581E-006  0.999999999999432       2.081668171172169E-015
```

The components of the unit thrust vector of the TLI impulsive maneuver are displayed in both the Earth-centered-inertial (ECI) and radial, tangential and normal (RTN) coordinate systems.

This section of the program output is created after the B-plane targeting problem has been solved. It includes a summary of the solution, the TLI delta-v vector and magnitude, the final B-plane coordinates and the orbital elements and state vector of the incoming hyperbola.

```
======================
optimal n-body solution
======================

descending transfer

transfer time        106.519756112248       hours

trans-lunar injection delta-v vector and magnitude
(geocentric Earth mean equator and equinox J2000)
-------------------------------------------------

deltav-x             2685.47702093217       meters/second
deltav-y            -1485.54512251347       meters/second
deltav-z            -629.489861744947       meters/second

deltav               3132.87226471460       meters/second

eci unit thrust vector

 0.857193269951853      -0.474179920849342      -0.200930586553070

rtn unit thrust vector

 3.929610390119714E-004  0.999999922789293      -1.740973091191034E-006

time and conditions prior to TLI
(geocentric Earth mean equator and equinox J2000)
-------------------------------------------------

UTC calendar date       September 15, 2008

UTC time                09:54:55.649

UTC Julian date         2454724.91314409

TDB time                09:56:00.832

TDB Julian date         2454724.91389851

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.656345630000D+04    0.716042246482D-16    0.285000000000D+02    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.353245173740D+03    0.245118791050D+03    0.245118791050D+03    0.146996629514D+01

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.335780398217D+04    -.487156385281D+04    -.284112242738D+04    0.656345630000D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.668164145782D+01    -.369300013531D+01    -.156450714128D+01    0.779296254099D+01

time and conditions after TLI
(geocentric Earth mean equator and equinox J2000)
-------------------------------------------------

UTC calendar date       September 15, 2008
```

UTC time 09:54:55.649

UTC Julian date 2454724.91314409

TDB time 09:56:00.832

TDB Julian date 2454724.91389851

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.191022462974D+06	0.965640395831D+00	0.285000120341D+02	0.245105601643D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.353245228119D+03	0.131416178811D-01	0.245118743261D+03	0.230799647130D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.335780398217D+04	-.487156385281D+04	-.284112242738D+04	0.656345630000D+04

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.936711847875D+01	-.517854525782D+01	-.219399700303D+01	0.109258346332D+02

time and conditions at lunar closest approach
(geocentric Earth mean equator and equinox of J2000)

UTC calendar date September 19, 2008

UTC time 20:26:06.771

UTC Julian date 2454729.35146726

TDB time 20:27:11.954

TDB Julian date 2454729.35222168

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.226037830064D+06	0.262793126648D+01	0.901673932456D+02	0.161060207106D+03

raan (deg)	true anomaly (deg)	arglat (deg)
0.231778348616D+03	0.354652795765D+03	0.155713002871D+03

rx (km)	ry (km)	rz (km)	rmag (km)
0.207825892551D+06	0.264611286018D+06	0.151828443552D+06	0.369137658028D+06

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.431679155458D+00	0.539374509368D+00	-.185628181220D+01	0.198067006867D+01

time and conditions at lunar closest approach
(selenocentric lunar mean equator & IAU node of epoch)

UTC calendar date September 19, 2008

UTC time 20:26:06.771

UTC Julian date 2454729.35146726

TDB time 20:27:11.954

TDB Julian date 2454729.35222168

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.683169467107D+04	0.126904013040D+01	0.899999999599D+02	0.140405474608D+03

raan (deg)	true anomaly (deg)	arglat (deg)
0.146641367110D+03	0.360000000000D+03	0.140405474608D+03

rx (km)	ry (km)	rz (km)	rmag (km)

```
 0.118297039598D+04    -.778800401735D+03    0.117144998570D+04    0.183800002517D+04

     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 0.130967217408D+01    -.862213640865D+00    -.189576658616D+01    0.246020009659D+01

b-plane coordinates of incoming hyperbola
(selenocentric lunar mean equator & IAU node of epoch)
-----------------------------------------------------

b-magnitude          5337.74803144691       kilometers
b dot r             -5337.74803144691       kilometers
b dot t             3.733693318963560E-006  kilometers
theta                270.000000040078       degrees
v-infinity           847.145240428195       meters/second
r-periapsis          1838.00002516655       kilometers
decl-asy             1.59325896718781       degrees
rasc-asy             326.641367108400       degrees

selenocentric flight path angle  -1.078655371150484E-011  degrees

-----------------------------------------------------
coordinates of the moon at closest approach
(geocentric Earth mean equator and equinox of J2000)
-----------------------------------------------------

UTC calendar date       September 19, 2008

UTC time                20:26:06.771

UTC Julian date         2454729.35146726

TDB time                20:27:11.954

TDB Julian date         2454729.35222168
     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.387057904331D+06    0.469260710975D-01    0.274682409204D+02    0.654909165021D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.352456394178D+03    0.357084293759D+03    0.625752102615D+02    0.665690548982D+03

     rx (km)                ry (km)               rz (km)               rmag (km)
 0.206588660958D+06    0.265719415783D+06    0.151041328247D+06    0.368916204034D+06

     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 -.880138031538D+00    0.553045427398D+00    0.224951785729D+00    0.106353444008D+01

declination         24.1683467556695       degrees

right ascension     52.1360342116955       degrees
```

Verification of the solution

After the n-body optimization finishes, the software will verify the solution by numerically integrating the equations of motion using a Runge-Kutta-Fehlberg 7(8) numerical method. The integration starts with the position and velocity vector of the trans-lunar transfer orbit immediately after the TLI impulsive maneuver and propagates to closest approach to the moon predicted by the software.

The following is the verification summary for this example.

```
==================================
verification of the n-body solution
==================================

time and conditions at lunar closest approach
(geocentric Earth mean equator and equinox of J2000)
```

```
-------------------------------------------------------

UTC calendar date       September 19, 2008

UTC time                20:26:06.771

UTC Julian date         2454729.35146726

TDB time                20:27:11.954

TDB Julian date         2454729.35222168

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.226037824690D+06    0.262793130505D+01    0.901673927406D+02    0.161060207167D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.231778348844D+03    0.354652795710D+03    0.155713002877D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.207825892558D+06    0.264611286027D+06    0.151828443521D+06    0.369137658025D+06

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.431679141547D+00    0.539374522535D+00    -.185628182291D+01    0.198067007926D+01

time and conditions at lunar closest approach
(selenocentric lunar mean equator & IAU node of epoch)
-------------------------------------------------------

UTC calendar date       September 19, 2008

UTC time                20:26:06.771

UTC Julian date         2454729.35146726

TDB time                20:27:11.954

TDB Julian date         2454729.35222168

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.683169495857D+04    0.126904011717D+01    0.899999999777D+02    0.140405474793D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.146641367130D+03    0.736558247354D-06    0.140405475530D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.118297040379D+04    -.778800405836D+03    0.117144995460D+04    0.183800001211D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.130967216081D+01    -.862213632161D+00    -.189576660132D+01    0.246020009815D+01

b-plane coordinates of incoming hyperbola
(selenocentric lunar mean equator & IAU node of epoch)
-------------------------------------------------------

b-magnitude          5337.74810923454       kilometers
b dot r             -5337.74810923454       kilometers
b dot t             2.076305861464789E-006  kilometers
theta                270.000000022287       degrees
v-infinity           847.145222603009       meters/second
r-periapsis          1838.00001210839       kilometers
decl-asy             1.59325954678117       degrees
rasc-asy             326.641367129470       degrees

selenocentric flight path angle   4.119460026174752E-007  degrees
```

Technical discussion

This section provides additional details about the numerical algorithms used in this computer program. The computational methods discussed here include solving the two body Lambert problem, the method used for propagating the spacecraft's geocentric trajectory, the algorithm used for targeting to the B-plane, and the geocentric-to-selenocentric coordinate transformation.

Nonlinear programming problem

A trajectory optimization problem can be described by a system of *dynamic variables*

$$\mathbf{z} = \begin{bmatrix} \mathbf{y}(t) \\ \mathbf{u}(t) \end{bmatrix}$$

consisting of the *state variables* $\mathbf{y}$ and the *control variables* $\mathbf{u}$ for any time *t*. In this discussion vectors are denoted in bold.

The system dynamics are defined by a vector system of ordinary differential equations called the *state equations* that can be represented as follows:

$$\dot{\mathbf{y}} = \frac{d\mathbf{y}}{dt} = \mathbf{f}\left[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t\right]$$

where **p** is a vector of problem *parameters* that is not time dependent.
The initial dynamic variables at time t_0 are defined by $\boldsymbol{\psi}_0 \equiv \boldsymbol{\psi}\left[\mathbf{y}(t_0), \mathbf{u}(t_0), t_0\right]$ and the terminal conditions at the final time t_f are defined by $\boldsymbol{\psi}_f \equiv \boldsymbol{\psi}\left[\mathbf{y}(t_f), \mathbf{u}(t_f), t_f\right]$. These conditions are called the *boundary values* of the trajectory problem.

The problem may also be subject to *path constraints* of the form $\mathbf{g}\left[\mathbf{y}(t), \mathbf{u}(t), t\right] = 0$.

For any mission time *t* there are also simple bounds on the state variables

$$\mathbf{y}_l \le \mathbf{y}(t) \le \mathbf{y}_u$$

the control variables

$$\mathbf{u}_l \le \mathbf{u}(t) \le \mathbf{u}_u$$

and the problem parameters

$$\mathbf{p}_l \le \mathbf{p}(t) \le \mathbf{p}_u$$

The basic nonlinear programming problem (NLP is to determine the control vector history and problem parameters that minimize the scalar performance index or objective function given by

$$J = \phi\left[\mathbf{y}(t_0), t_0, \mathbf{y}(t_f), t_f, \mathbf{p}\right]$$

while satisfying all the user-defined mission constraints.

During the two-body trajectory optimization, the control variables are the TLI calendar date and the true anomaly of the TLI maneuver. For the numerical integration optimization, the control variables consist of the TLI calendar date, the RAAN of the park orbit, the true anomaly of the TLI maneuver, and the Cartesian components of the TLI delta-v vector.

For both types of optimization, the objective function or performance index is the scalar magnitude of the TLI impulsive delta-v.

In addition to the bounds on the TLI calendar date mentioned earlier, the true anomaly during the two-body optimization is bounded according to

$$-180^\circ \le \theta \le +180^\circ$$

During the second part of the trajectory optimization, the RAAN and true anomaly bounds are

$$\Omega_{TB} - 30^\circ \le \Omega \le \Omega_{TB} + 30^\circ$$

$$\theta_{TB} - 30^\circ \le \theta \le \theta_{TB} + 30^\circ$$

where Ω_{TB} and θ_{TB} are the RAAN and true anomaly found during the two-body optimization. The bounds on the components of the TLI delta-v are given by

$$-\left(|\Delta\mathbf{v}| + 0.1|\Delta\mathbf{v}|\right) \le \Delta\mathbf{v}_{x,y,z} \le +\left(|\Delta\mathbf{v}| + 0.1|\Delta\mathbf{v}|\right)$$

where $|\Delta\mathbf{v}|$ is the scalar magnitude of the two-body TLI delta-v vector.
The final boundary conditions are the B-plane coordinates of the incoming selenocentric hyperbola.

Modeling the spacecraft's trajectory

The spacecraft's orbital motion is modeled with respect to the Earth mean equator and equinox of J2000 (EME2000) coordinate system. Please consult *Appendix G – Aerospace Trajectory Coordinates and Time Systems* for information about this system. Numerical methods for solving Lambert's problem can be found in *Appendix I – Numerical Solutions of Lambert's Problem*.

In this computer program the heliocentric coordinates of the sun and moon are based on the JPL Development Ephemeris DE421. These coordinates are provided in the Earth mean equator and equinox of J2000 coordinate system (EME2000). The name of this binary date file is `de421.bin`, and it must reside in the same directory as the `tlto.exe` computer program.

Park orbit RAAN

For a given TLI injection time, there are two possible locations on the initial park orbit at which to perform the propulsive maneuver. One opportunity occurs during the ascending part of the park orbit and the other during the descending motion. The park orbit RAAN Ω_p at these two locations can be determined from spherical trigonometry relationships involving the park orbit inclination and the right ascension and declination of the moon at encounter.

ascending $$\Omega_p = -180^\circ + \alpha_m + \sin^{-1}\left(\tan\delta_m / \tan i_p\right)$$

descending $$\Omega_p = \alpha_m - \sin^{-1}\left(\tan\delta_m / \tan i_p\right)$$

where

$\alpha_m =$ right ascension of the moon at encounter
$\delta_m =$ declination of the moon at encounter
$i_p =$ park orbit inclination

These opportunities are valid whenever $|\delta_m| \le i_p$.

B-plane targeting

The software solves the B-plane targeting problem by minimizing the delta-v vector at the TLI while satisfying two nonlinear *equality constraint* equations. These constraint equations are the differences between components of the *required* B-plane and the B-plane components *predicted* by the software.

Given the user-defined closest approach radius r_{ca} and orbital inclination *i*, and the incoming v-infinity magnitude v_∞ and the right ascension α_∞ and declination δ_∞ of the incoming asymptote vector at the moment of closest approach, the following series of equations can be used to determine the *required* B-plane target components

$$\mathbf{B} \bullet \mathbf{T} = b_t \cos\theta$$

$$\mathbf{B} \bullet \mathbf{R} = b_t \sin\theta$$

where

$$b_t = \sqrt{\frac{2\mu r_{ca}}{v_\infty^2} + r_{ca}^2} = r_{ca}\sqrt{1 + \frac{2\mu}{r_{ca} v_\infty^2}}$$

and

$$\cos\theta = \frac{\cos i}{\cos\delta_\infty} \qquad \sin\theta = -\sqrt{1-\cos^2\theta} \quad \rightarrow \quad \theta = \tan^{-1}(\sin\theta, \cos\theta)$$

$$\sin\delta_\infty = |\hat{\mathbf{s}} \times \hat{\mathbf{z}}| = \sqrt{s_x^2 + s_y^2} \qquad \hat{\mathbf{z}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

The arrival asymptote unit vector $\hat{\mathbf{S}}$ is given by

$$\hat{\mathbf{S}} = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

where δ_∞ and α_∞ are the declination and right ascension of the asymptote of the incoming hyperbola. Please consult *Appendix D – B-Plane Geometry, Coordinates and Targeting* for the definition of the B-plane coordinate system along with pertinent equations.

Important note!!

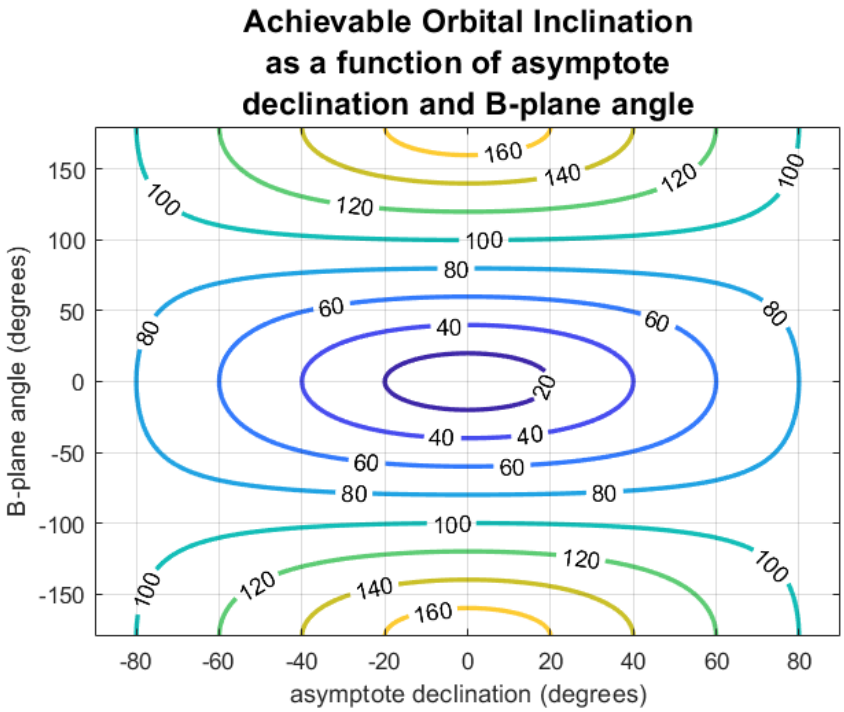
This technique is valid for lunar orbit inclinations that satisfy $|i| > |\delta_\infty|$.

If this inequality is not satisfied, the software will print the following error message

```
b-plane targeting error!!
|inclination| must be > |asymptote declination|
```

It will also display the actual declination of the asymptote and stop. The user should then edit the input file, include a valid orbital inclination and restart the simulation.

The relationship between orbital inclination *i*, B-plane angle θ and asymptote declination δ of an incoming or outgoing hyperbola is given by $\cos i = \cos\theta\cos\delta$. The following is a contour plot illustrating the achievable inclination as a function of B-plane angle and declination.



Targeting to the selenocentric periapsis radius and orbital inclination

For this targeting option, the equality constraints enforced by the nonlinear programming algorithm are

$$r_p - r_{ca} = 0 \qquad \cos i - \hat{\mathbf{h}}_z = 0$$

where r_p and i are the user-defined periapsis radius and selenocentric orbital inclination, respectively. In the second equation $\hat{\mathbf{h}}_z$ is the z-component of the spacecraft's unit angular momentum vector at closest approach to the moon.

For both types of targeting techniques, closest approach is determined during the numerical integration of the spacecraft equations of motion by finding the time since TLI at which the selenocentric flight path angle is zero. This mission constraint is computed as follows

$$\sin\gamma = \left(\frac{\mathbf{r} \bullet \mathbf{v}}{|\mathbf{r} \bullet \mathbf{v}|} \right) \approx 0$$

where **r** and **v** are the spacecraft's moon-centered position and velocity vectors, respectively.

Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `tlto` software.

The simulation summary screen display contains the following information.

`transfer time` = total time from the TLI maneuver to closest approach at the moon

`UTC time` = simulation event time on the UTC time scale

`TDB time` = simulation event time on the TDB time scale

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (min) = orbital period in minutes

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

delta-vx = x-component of the TLI impulsive velocity vector in meters/second

delta-vy = y-component of the TLI impulsive velocity vector in meters/second

delta-vz = z-component of the TLI impulsive velocity vector in meters/second

delta-v = scalar magnitude of the TLI maneuver in meters/seconds

b-magnitude = magnitude of the b-plane vector

b dot r = dot product of the b-vector and r-vector

b dot t = dot product of the b-vector and t-vector

theta = orientation of the b-plane vector

v-infinity = magnitude of incoming v-infinity vector

r-periapsis = periapsis of incoming hyperbola

decl-asy = declination of incoming v-infinity vector

rasc-asy = right ascension of incoming v-infinity vector

The comma-separated-variable disk file contains the following information:

time (sec) = simulation time since TLI maneuver in seconds

time (min) = simulation time since TLI maneuver in minutes

time (hr) = simulation time since TLI maneuver in hours

re2sc-x (km) = x-component of the spacecraft's geocentric position vector in kilometers

re2sc-y (km) = y-component of the spacecraft's geocentric position vector in kilometers

re2sc-z (km) = z-component of the spacecraft's geocentric position vector in kilometers

re2sc-mag (km) = the spacecraft's geocentric radius in kilometers

ve2sc-y (km/sec) = y-component of the spacecraft's geocentric velocity vector in kilometers per second

ve2sc-z (km/sec) = z-component of the spacecraft's geocentric position vector in kilometers per second

ve2sc-mag (km/sec) = the spacecraft's geocentric speed in kilometers per second

rm2sc-x (km) = x-component of the spacecraft's selenocentric position vector in kilometers

rm2sc-y (km) = y-component of the spacecraft's selenocentric position vector in kilometers

rm2sc-z (km) = z-component of the spacecraft's selenocentric position vector in kilometers

rm2sc-mag (km) = the spacecraft's selenocentric radius in kilometers

vm2sc-y (km/sec) = y-component of the spacecraft's selenocentric velocity vector in kilometers per second

vm2sc-y (km/sec) = y-component of the spacecraft's selenocentric velocity vector in kilometers per second

vm2sc-z (km/sec) = z-component of the spacecraft's selenocentric position vector in kilometers per second

vm2sc-mag (km/sec) = the spacecraft's selenocentric speed in kilometers per second

re2m-x (km) = x-component of the moon's geocentric position vector in kilometers

re2m-y (km) = y-component of the moon's geocentric position vector in kilometers

re2m-z (km) = z-component of the moon's geocentric position vector in kilometers

re2m-mag (km) = the moon's geocentric radius in kilometers

sma-geo (km) = geocentric semimajor axis of the spacecraft in kilometers

ecc-geo = geocentric orbital eccentricity of the spacecraft(non-dimensional)

incl-geo (deg) = geocentric orbital inclination of the spacecraft in degrees

argper-geo (deg) = geocentric argument of perigee of the spacecraft in degrees

raan-geo (deg) = geocentric right ascension of the ascending node of the spacecraft in degrees

tanom-geo (deg) = geocentric true anomaly of the spacecraft in degrees

sma-sel (km) = selenocentric semimajor axis of the spacecraft in kilometers

ecc-sel = selenocentric orbital eccentricity of the spacecraft(non-dimensional)

incl-sel (deg) = selenocentric orbital inclination of the spacecraft in degrees

argper-sel (deg) = selenocentric argument of perigee of the spacecraft in degrees

raan-sel (deg) = selenocentric right ascension of the ascending node of the spacecraft in degrees

tanom-sel (deg) = selenocentric true anomaly of the spacecraft in degrees

The geocentric coordinates of the spacecraft and the TLI delta-v components are with respect to the Earth mean equator and equinox of J2000 (EME2000) coordinate system. The selenocentric coordinates are with respect to the mean lunar equator and IAU node of epoch coordinate system. Additional information about the selenocentric coordinate system implemented in this computer program can be found in Appendix G – Aerospace Trajectory Coordinates and Time Systems.

"Lunar Trajectories", NASA TN D-866, August 1961.

"Earth-Moon Trajectories", JPL Technical Report No. 32-503, May 1, 1964.

"Three-Dimensional Lunar Trajectories", V. A. Egorov, Mechanics of Space Flight Series, Israel Program for Scientific Translations, Jerusalem 1969.

"Circumlunar Trajectory Calculations", MIT Instrumentation Laboratory Report R-353, April 1962.

"Optimal Low Thrust Trajectories to the Moon", John T. Betts and Sven O. Erb, *SIAM Journal on Applied Dynamical Systems*, Vol. 2, No. 2, pp. 144-170, 2003.

"Integrated Algorithm for Lunar Transfer Trajectories Using a Pseudostate Technique", R. V. Ramanan, AIAA *Journal of Guidance, Control and Dynamics*, Vol. 25, No. 5, September-October 2002, pp. 946-952.

"Injection Conditions for Lunar Trajectories", R. Kolenkiewicz and W. Putney, NASA TM X-55390, November 1965.

"Coplanar Three-Body Trans-Earth Lunar Trajectory Simulation Methodology", H. Ikawa, AIAA 88-0381, AIAA 26th Aerospace Sciences Meeting, Reno, Nevada, January 11-14, 1988.

"Lunar Constants and Models Document", JPL D-32296, September 23, 2005.

CMATO 10 – Finite-burn Trans-lunar Trajectory Optimization

This *CMATO* application is a Fortran computer program named `tlto_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the classic finite-burn, trans-lunar injection (TLI) trajectory optimization problem. The software attempts to maximize the spacecraft mass at the end of the TLI propulsive maneuver while targeting to a user-defined periapsis radius and orbital inclination relative to the moon. Since the TLI is a continuous thrust maneuver, maximizing the spacecraft mass is equivalent to minimizing the propellant required for the maneuver.

The important features of this scientific simulation are

- single, continuous thrust propulsive maneuver followed by a coast to lunar close approach
- combination of modified equinoctial and cartesian equations of motion
- valid for circular and elliptical park orbits
- Earth J_2 gravity model and sun and moon point-mass gravity
- JPL DE421 lunar and solar ephemeris
- B-plane coordinates at lunar close approach

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, “Practical Methods for Optimal Control and Estimation Using Nonlinear Programming” by John. T. Betts, SIAM, 2010.

The `tlto_sos` software consists of Fortran routines that perform the following tasks.

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user’s problem and perform the optimization.

The `tlto_sos` software allows the user to select the type of initial guess, collocation method and other important algorithm control parameters.

Input file format and contents

The `tlto_sos` computer program is "data-driven" by a simple user-created text file. The following is a typical input or "simulation definition" file used by the software. In this discussion the actual input file contents are in *courier* font and all explanations are in times font. Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is <u>not</u> case sensitive but must be spelled correctly.

Please note that the fundamental time argument in this simulation is Barycentric Dynamical Time (TDB) which is the time argument of the DE421 ephemeris. Furthermore, the fundamental coordinate system is the Earth mean equator and equinox of J2000 (EME2000) described in *Appendix G – Aerospace Trajectory Coordinates and Time Systems*.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*********************************************
** finite-burn translunar trajectory optimization
** two phase, 3-body geocentric motion
** input file for tlto_sos
** tlto1.in - August 7, 2012
*********************************************
```

The software allows the user to specify an initial guess for the calendar date and time of the TLI maneuver and lower and upper bounds on the actual date found during the optimization process. For any guess for maneuver time t_{TLI} and user-defined lower and upper bounds Δt_l and Δt_u, the TLI maneuver time t is constrained as follows

$$t_{TLI} - \Delta t_l \le t \le t_{TLI} + \Delta t_u$$

For a fixed maneuver time, the lower and upper bounds should be set to zero.

The user inputs for the initial calendar date, TDB time, and search boundary flow this format.

```
initial calendar date (month, day, year)
10, 13, 2008

initial TDB time guess (hours, minutes, seconds)
0, 0, 0

lower bound for initial time guess (hours)
-24.0

upper bound for initial time guess (hours)
+24.0
```

The next three inputs define the initial spacecraft mass, the thrust magnitude and specific impulse of the propulsion system, respectively.

```
initial spacecraft mass (kilograms)
1000.0

thrust magnitude (newtons)
5000.0
```

```
specific impulse (seconds)
450.0
```

The type of propulsion initial guess is determined by the next integer input.

```
*******************************
type of propulsive initial guess
*******************************
1 = thrust duration
2 = delta-v
-------------------
2
```

For option 1, the next input is the user's initial guess for the magnitude of the maneuver delta-v.

```
initial guess for delta-v (meters/second)
3150.0
```

For option 2, the next three inputs are the user's initial guess for the thrust duration and lower and upper bounds for this duration.

```
initial guess for thrust duration (seconds)
3000.0

lower bound for thrust duration (seconds)
100.0

upper bound for thrust duration (seconds)
1000.0
```

The next series of inputs define the characteristics of the initial park orbit. The angular orbital elements should be with respect to the EME2000 coordinate system.

```
******************
initial park orbit
******************

semimajor axis (kilometers)
6563.3363

orbital eccentricity (non-dimensional)
0.0d0

orbital inclination (degrees)
28.5d0

argument of perigee (degrees)
0.0d0

right ascension of the ascending node (degrees)
282.787

true anomaly (degrees)
305.233
```

This next integer input allows the user to define the type of initial park orbit constraints to use during the simulation. Option 1 will constrain all elements of the park orbit except the true longitude to the values input by the user. Option 2 will constrain the semimajor axis, eccentricity, and orbital inclination. The RAAN and true longitude will be bounded.

```
****************************
park orbit constraint options
****************************
```

```
1 = constrain all initial orbital elements except true longitude
2 = constrained a, e, i; bounded raan & true longitude
-----------------------------------------------------
2
```

The next set of inputs defines the user's initial guess for the lunar transfer time, along with a lower and upper bound for the transfer time. The transfer time here refers to the time from burnout of the propulsive maneuver to the entrance to the lunar sphere-of-influence (SOI).

```
transfer time initial guess (hours)
96.0

transfer time lower bound (hours)
84.0d0

transfer time upper bound (hours)
108.0d0
```

The next two inputs define the periapsis radius and orbital inclination of the lunar trajectory relative to the moon. The inclination should be specified relative to the mean lunar equator.

```
-----------------------------------------
final lunar orbit characteristics
(lunar mean equator and IAU node of epoch)
-----------------------------------------

periapsis radius (kilometers)
1838.0

orbital inclination (degrees)
90.0
```

The next three inputs define the types of perturbations to include during the numerical solution of the spacecraft's equations of motion. The solar and lunar perturbations are modeled as point-mass bodies.

```
************************
trajectory perturbations
************************

include j2 gravity perturbation (1 = yes, 0 = no)
1

include solar perturbations (1 = yes, 0 = no)
1

include lunar perturbations (1 = yes, 0 = no)
1
```

The next program input is the user-defined radius of the lunar sphere-of-influence (SOI) used by the software during the trajectory optimization. Typical values for this parameter are between 64,000 and 20,000 kilometers.

```
-------------------------------------------------
radius of lunar sphere-of-influence (kilometers)
-------------------------------------------------
25000.0
```

This next input defines the type of initial guess to use. Please see the technical discussion section for information about how the first option is modeled. Option 2 requires either a binary restart file created from a previous run using either initial guess option 1 or an updated binary restart file. This feature is described in the next two sections.

```
*******************************
* initial guess/restart option *
*******************************
 1 = numerical integration
 2 = binary data file
---------------------
1
```

If the user elects to use a binary data file (option 2 above) for the initial guess, the following text input specifies the name of the file to use.

```
name of initial guess/restart input data file
tlto1.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess option 1, the software will create a binary restart file. For initial guess option 2, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*****************************
* binary restart file option *
*****************************

create/update binary restart data file (yes or no)
no
```

This next input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by the *Sparse Optimization Suite*. Options 2 and 3 allow the user to specify either the number of nodes or time step size used to create the data file.

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = OC-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this next input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
1
```

The name of the solution data file is defined in this next line.

```
name of solution output file
tlto1.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
```

```
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
------------------------------
1
```

The next integer defines the number of initial grid points to use in the collocation modeling of the propulsive maneuver (phase 1) and the lunar coast phase (phase 2).

```
number of grid points in phase 1 (TLI thrust maneuver)
10

number of grid points in phase 2 (coast to lunar encounter)
75
```

The next input defines the relative error in the objective function. A value of `1.0d-5` is recommended.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations. A value of `1.0d-7` is recommended.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
5000
```

The level of output from the NLP algorithm is controlled with the following integer input. Additional information about these algorithm items can be found in the *Sparse Optimization Suite* user's manual.

```
**************************
sparse NLP iteration output
--------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output from the optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
```

```
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the differential equation algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
****************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of inputs allow the reading and writing of *SOS* configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `tlto_sos` software. Please consult *Appendix J – Sparse Optimization Suite Configuration File* for additional information about this file.

```
************************************
* optimal control configuration options
************************************

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
tlto1_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
tlto1_config.txt
```

Optimal control solution

The following is the program output created by the `tlto_sos` scientific simulation for this example. This display summarizes the characteristics of the optimized TLI maneuver and the orbital transfer conditions before and after the propulsive maneuver. It also displays the conditions at the lunar SOI, the closest approach to the moon, and the corresponding B-plane characteristics. The delta-v magnitude is determined from a cubic spline integration of the thrust acceleration evaluated at the grid points determined by the *Sparse Optimization Suite*.

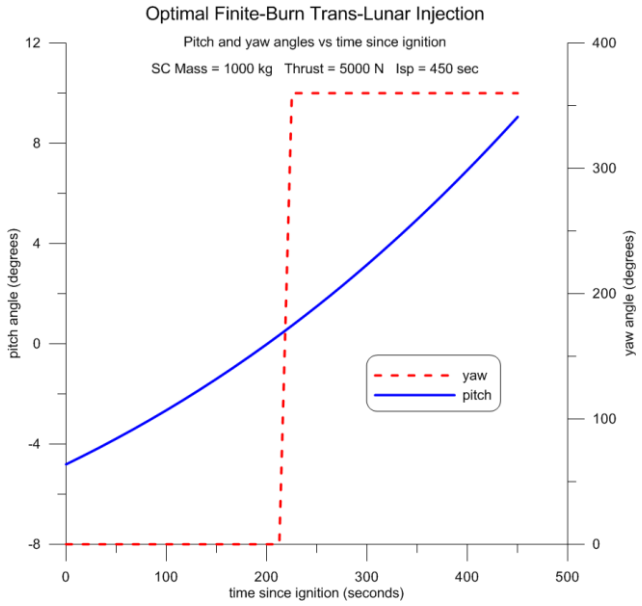
```
----------------
program tlto_sos
----------------

input data file ==> tlto1.in

numerical integration initial guess

a, e, i constrained; bounded RAAN & true longitude

-----------------------------------------------------
park orbit initial conditions
(geocentric - Earth mean equator and equinox of J2000)
-----------------------------------------------------

calendar date           October 12, 2008

TDB time                16:45:56.642

TDB Julian date         2454752.19857224

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.656333630000D+04    0.337656355758D-15    0.285000000001D+02    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.287195861707D+03    0.286043048287D+03    0.286043048287D+03    0.146992598213D+01

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.475930402705D+04    -.337158742648D+04    -.300978517350D+04    0.656333630000D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.402228337545D+01    -.659519173681D+01    0.102764678874D+01    0.779303378153D+01

-----------------------------------------------------
time and conditions at end of TLI finite burn
(geocentric - Earth mean equator and equinox of J2000)
-----------------------------------------------------

calendar date           October 12, 2008

TDB time                16:53:26.805

TDB Julian date         2454752.20378247

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.188605021797D+06    0.964966562528D+00    0.285136491860D+02    0.303164578156D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.287146403096D+03    0.184559477276D+02    0.321620525884D+03    0.226432289601D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.196718305280D+04    -.616802312790D+04    -.200908455122D+04    0.677869745626D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.843662330982D+01    -.568110244384D+01    0.346971081493D+01    0.107466474302D+02

final mass           489.956582872968       kilograms

propellant mass      510.043417127032       kilograms

thrust duration      450.163554891133       seconds

delta-v              3148.40228435593       meters/second
```

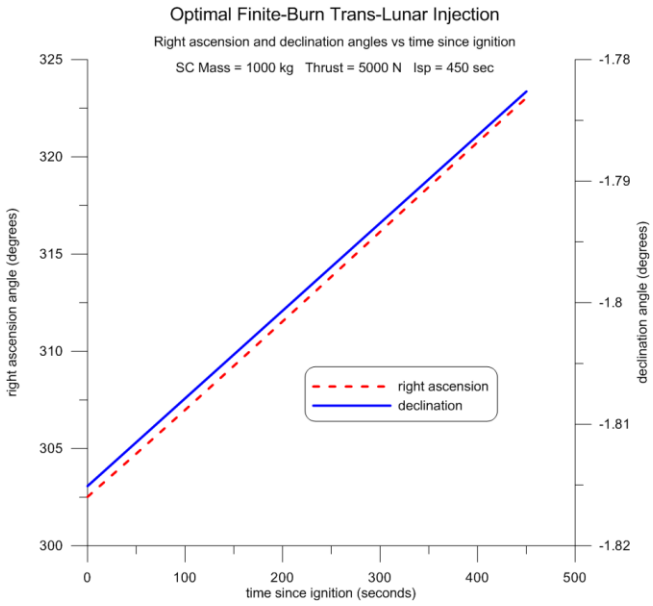
```
-------------------------------------------------------
time and conditions at beginning of trans-lunar coast
(geocentric - Earth mean equator and equinox of J2000)
-------------------------------------------------------

calendar date           October 12, 2008

TDB time                16:53:26.805

TDB Julian date         2454752.20378247

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.188605022221D+06    0.964966562606D+00    0.285136491854D+02    0.303164578158D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.287146403096D+03    0.184559477263D+02    0.321620525884D+03    0.226432290364D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.196718305288D+04    -.616802312802D+04    -.200908455121D+04    0.677869745639D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.843662330994D+01    -.568110244389D+01    0.346971081493D+01    0.107466474303D+02

-------------------------------------------------------
time and conditions at lunar sphere-of-influence
(geocentric - Earth mean equator and equinox of J2000)
-------------------------------------------------------

calendar date           October 16, 2008

TDB time                19:36:15.566

TDB Julian date         2454756.31684683

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.182765112444D+06    0.991136395952D+00    0.788292866378D+02    0.336687837478D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.235996970525D+03    0.179356842529D+03    0.156044680006D+03    0.215997351927D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.208245444983D+06    0.257873126624D+06    0.143942931031D+06    0.361364472776D+06

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.792384820463D-01    0.131480006199D+00    -.396969018687D-01    0.158560944387D+00

-------------------------------------------------------
time and conditions at lunar sphere-of-influence
(selenocentric lunar mean equator & IAU node of epoch)
-------------------------------------------------------

calendar date           October 16, 2008

TDB time                19:36:15.566
TDB Julian date         2454756.31684683

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.619130066766D+04    0.129628416983D+01    0.902141274019D+02    0.315630146925D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.325541802794D+03    0.230099522693D+03    0.185729669618D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.205052179728D+05    0.140821422237D+05    -.249585778317D+04    0.249999999959D+05

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.895329539381D+00    -.614047330240D+00    -.737580629172D-01    0.108816789132D+01
```

```
------------------------------------------------------
time and conditions at lunar closest approach
(selenocentric lunar mean equator & IAU node of epoch)
------------------------------------------------------

calendar date           October 17, 2008

TDB time                01:04:40.769

TDB Julian date         2454756.54491631

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.615478109923D+04    0.129862962957D+01    0.900000000013D+02    0.315506025045D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.325500870882D+03    0.895644198300D-11    0.315506025045D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.108051471785D+04    -.742593009982D+03    -.128813336463D+04    0.183799999974D+04

     vx (kps)               vy (kps)              vz (kps)             vmag (kps)
 0.143020212900D+01    -.982918683482D+00    0.176632564306D+01    0.247618931939D+01

------------------------------------------------------
b-plane coordinates of incoming hyperbola
(selenocentric lunar mean equator & IAU node of epoch)
------------------------------------------------------

b-magnitude          5099.33518378722       kilometers
b dot r              5099.33518378722
b dot t             -1.135427112330945E-007
theta                90.0000000012758       degrees
v-infinity           892.515554352604       meters/second
r-periapsis          1837.99999974034       kilometers
decl-asy            -4.85167997302203       degrees
rasc-asy             325.500870881723       degrees

selenocentric flight path angle    5.066484611301138E-012  degrees


TLI to SOI duration                 98.8385901041329       hours

TLI to closest approach duration    104.312257602811       hours

------------------------------------------------------
coordinates of the moon at closest approach
(geocentric - Earth mean equator and equinox of J2000)
------------------------------------------------------

calendar date           October 17, 2008

TDB time                01:04:40.769

TDB Julian date         2454756.54491631

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.390199354514D+06    0.675885534905D-01    0.272667968985D+02    0.638187138880D+02

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.352327432109D+03    0.357204061295D+03    0.610227751834D+02    0.673811315346D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.212470264207D+06    0.256867171511D+06    0.145825944378D+06    0.363853765991D+06

     vx (kps)               vy (kps)              vz (kps)             vmag (kps)
 -.877338109248D+00    0.585467570386D+00    0.238679391860D+00    0.108141679570D+01
```

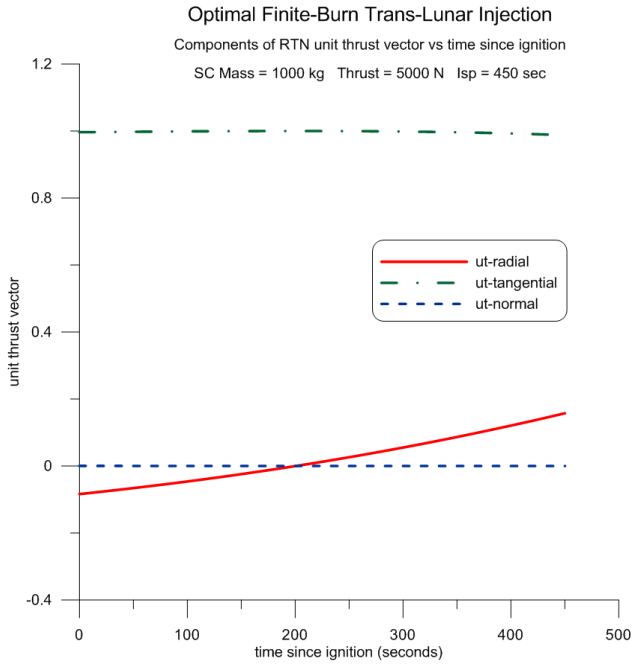
```
declination          23.6270577371795        degrees

right ascension      50.4038062239193        degrees
```

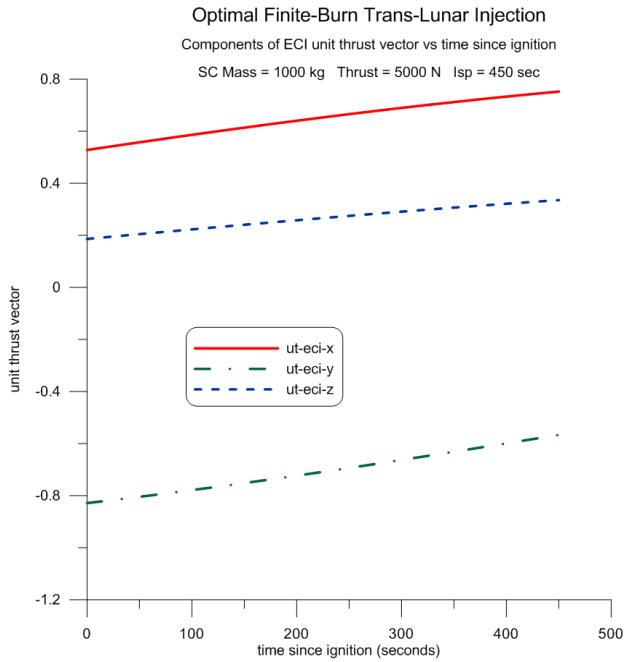
The following plots illustrate the behavior of the pitch, yaw, right ascension, and declination angles during the propulsive maneuver.



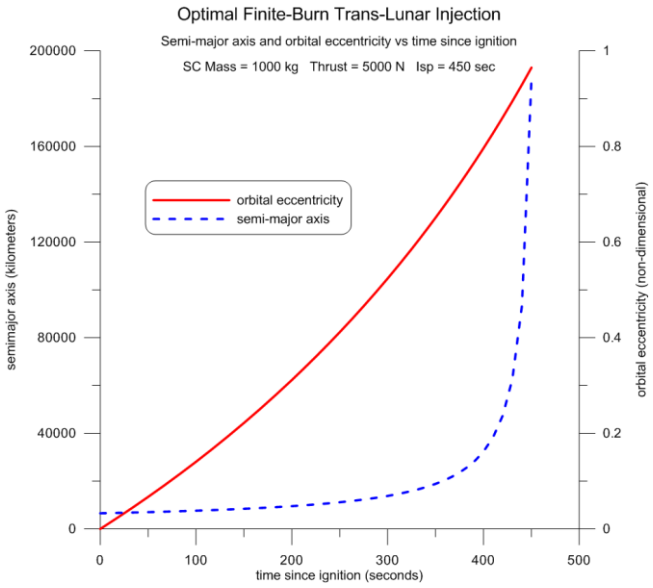


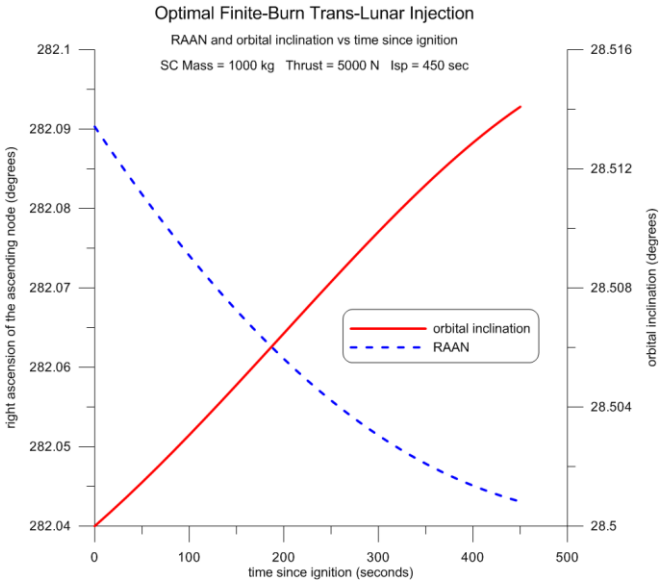
These next two plots illustrate the behavior of the individual components of the radial-tangential-normal (RTN) and Earth-centered-inertial (ECI) unit thrust vector during the maneuver.





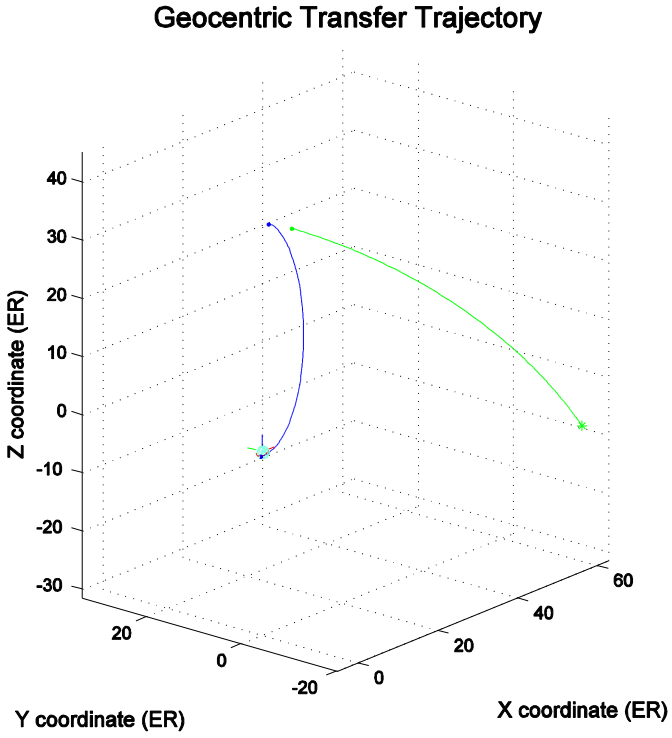
The final two plots display the orbital evolution of the semimajor axis, eccentricity, RAAN and orbital inclination during the propulsive maneuver.



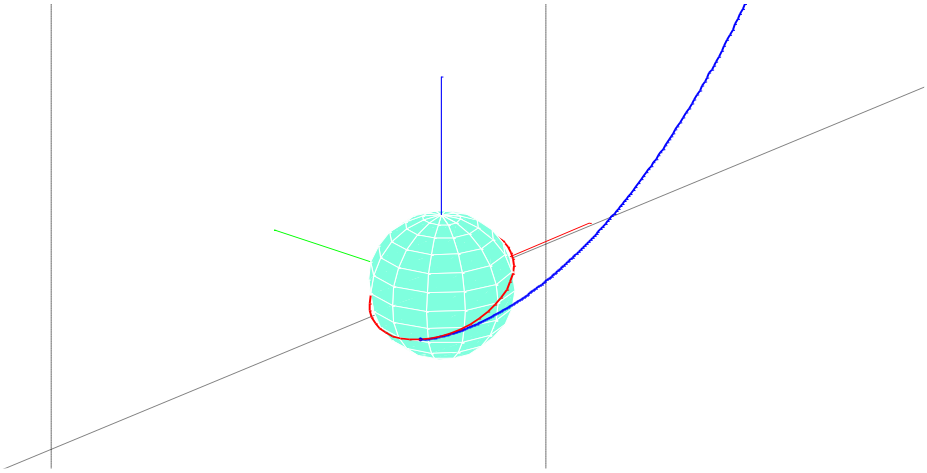


The `tlto_sos` software will create three comma-separated-variable (csv) output files. The first file is named `parkorb.csv` and contains the geocentric, EME 2000 state vector of the park orbit prior to the orbital maneuver. The second file contains the state vectors and orbital elements of the geocentric transfer orbit with the name specified by the user in the main input data file. The third file is named `soiorb.csv` and contains the state vector of the selenocentric orbit at the lunar sphere-of-influence.

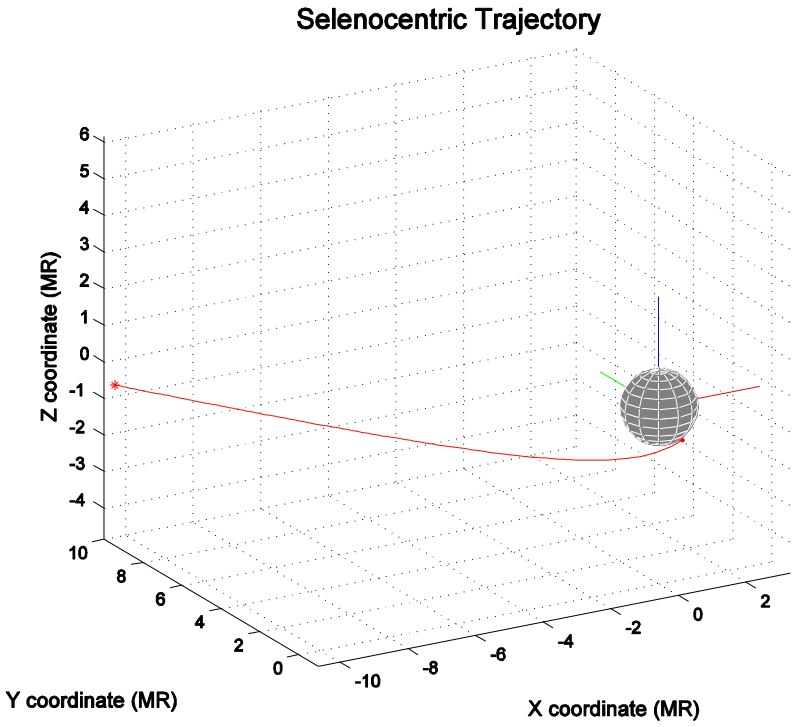
The following is the transfer trajectory from burnout of the TLI maneuver to the lunar SOI for this example. The park orbit trajectory is red, the lunar transfer is blue and the moon's orbit is green. The location of the moon at the time of the TLI maneuver is marked with a green asterisk. The coordinates are in the units of Earth radius (ER). This display is also labeled with a geocentric, inertial coordinate system. The x-axis of this system is red, the y-axis is green and the z-axis is blue.



The following is a “zoomed” plot of the park orbit and initial portion of the lunar transfer trajectory.



This final plot illustrates the selenocentric hyperbola trajectory within the user-defined lunar sphere-of-influence (SOI). The coordinate units are lunar radii (MR). The entry into the SOI is marked with an asterisk and the location of the periapsis or lunar closest approach is marked with a red dot. This display is labeled with a selenocentric, inertial coordinate system (lunar mean equator and IAU node of epoch). The x-axis is red, the y-axis is green and the z-axis is blue.



Verification of the optimal control solution

The optimal control solution determined by the software can be verified by numerically integrating the orbital equations of motion with the computed initial park orbit conditions and the optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the optimal unit thrust vector solution. This part of the `tlto_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is the program output of the final solution computed using this *explicit* numerical integration method.

```
========================================
verification of optimal control solution
========================================

------------------------------------------------------
time and conditions at end of TLI finite burn
(geocentric - Earth mean equator and equinox of J2000)
------------------------------------------------------

calendar date           October 12, 2008

TDB time                16:53:26.805

TDB Julian date         2454752.20378247

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.188605010467D+06    0.964966560164D+00    0.285136492601D+02    0.303164576336D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.287146402598D+03    0.184559496421D+02    0.321620525978D+03    0.226432269197D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.196718311958D+04    -.616802319271D+04    -.200908457735D+04    0.677869754236D+04

     vx (kps)               vy (kps)              vz (kps)             vmag (kps)
 0.843662313135D+01    -.568110260966D+01    0.346971074385D+01    0.107466473548D+02
final mass           489.956582873049       kilograms

propellant mass      510.043417126951       kilograms

thrust duration      450.163554891133       seconds

delta-v              3148.39874170068       meters/second

------------------------------------------------------
time and conditions at lunar sphere-of-influence
(selenocentric lunar mean equator & IAU node of epoch)
------------------------------------------------------

calendar date           October 16, 2008

TDB time                19:36:15.566

TDB Julian date         2454756.31684683

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.619130231760D+04    0.129628437873D+01    0.902133674706D+02    0.315630131786D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.325541849903D+03    0.230099543782D+03    0.185729675568D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.205052416109D+05    0.140820935004D+05    -.249585970474D+04    0.249999921308D+05

     vx (kps)               vy (kps)              vz (kps)             vmag (kps)
 0.895329454928D+00    -.614047375945D+00    -.737581134894D-01    0.108816785106D+01

------------------------------------------------------
time and conditions at lunar closest approach
(selenocentric lunar mean equator & IAU node of epoch)
------------------------------------------------------

calendar date           October 17, 2008
```

```
TDB time                01:04:40.769

TDB Julian date         2454756.54491631

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.615478273561D+04    0.129862987923D+01    0.899992378203D+02    0.315506007238D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.325500917242D+03    0.367010839840D-03    0.315506374249D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.108051327414D+04    -.742611522507D+03    -.128812679311D+04    0.183800202505D+04

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.143021191825D+01    -.982895196640D+00    0.176632906232D+01    0.247618808960D+01

-------------------------------------------------------
b-plane coordinates of incoming hyperbola
(selenocentric lunar mean equator & IAU node of epoch)
-------------------------------------------------------

b-magnitude          5099.33894806607       kilometers
b dot r              5099.33894761164
b dot t             6.807811598068270E-002
theta                89.9992350795343       degrees
v-infinity           892.515435705324       meters/second
r-periapsis          1838.00202502428       kilometers
decl-asy            -4.85168448435964       degrees
rasc-asy             325.500852547647       degrees

selenocentric flight path angle    2.073457962401969E-004  degrees
```

Creating an initial guess

This section describes the algorithms used in `tlto_sos` to create an initial guess for the software. The software attempts to obtain problem feasibility before trying to solve the optimal control problem. If the software cannot get feasible, the user will have to provide a better initial guess.

The software allows the user to input either a delta-v magnitude or thrust duration initial guess. For a delta-v initial guess, the software estimates the thrust duration using the rocket equation. The user should also provide lower and upper bounds for the total thrust duration. These three inputs should be in units of seconds.

For trans-lunar injection from circular and near-circular orbits, the delta-v magnitude can be estimated from *CAMTO 9 – Impulsive Trans-Lunar Trajectory Optimization* which calculates the *impulsive* delta-v required to perform a TLI maneuver. It will also provide an initial guess for the park orbit right ascension of the ascending node (RAAN) and the park orbit true anomaly at the TLI impulse.

For the numerical integration initial guess option, the software models the maneuver using tangential thrusting. For this case, the unit thrust vector in the modified equinoctial frame is simply $\mathbf{u}_T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$. Please note that this type of steering method creates a *coplanar* initial guess that increases the spacecraft's orbital energy.

An estimate of the thrust duration can be determined from the following expression:

$$t_d = \frac{I_{sp} m_p g}{F} = \frac{m_p V_{ex}}{F}$$

The propellant mass required for a given ΔV is a function of the initial mass of the spacecraft and the exhaust velocity as

$$m_p = m_i \left(1 - e^{\frac{-\Delta V}{V_{ex}}} \right)$$

In these equations

$m_i =$ initial mass
$m_p =$ propellant mass
$V_{ex} =$ exhaust velocity = $g I_{sp}$
$I_{sp} =$ specific impulse
$\Delta V =$ impulsive velocity increment
$F =$ thrust
$g =$ acceleration of gravity

Since a finite burn maneuver will require a thrust duration longer than this impulsive estimate, the user can increase this value by about 10% and use it for the initial guess required by `tlto_sos`.

The initial guess for the *Sparse Optimization Suite* is created by numerically integrating the modified equinoctial equations of motion for the user-defined initial guess or rocket equation calculation for the orbital maneuver time.

Binary restart data files can also be used to initialize a `tlto_sos` simulation. A typical scenario is

1) create a binary restart file from a converged and optimized simulation
2) modify the original input file with slightly different spacecraft characteristics, propulsive parameters or perhaps final mission targets and/or constraints
3) use the previously created binary restart file as the initial guess for the new simulation

This technique works well provided the two simulations are not dramatically different. Sometimes it is necessary to make successive small changes in the mission definition and run multiples simulations to eventually reach the final desired solution.

Problem setup

This section describes several methods that can be used to create an initial guess for `tlto_sos`. The *Sparse Optimization Suite* attempts to obtain problem feasibility before trying to solve the optimal control problem. If the optimal control algorithm cannot find a feasible solution, the software will print warnings and the user will have to provide a better initial guess.

Point functions – initial orbit and mass constraints

The software allows the user to select one of the following initial orbit constraint options:

4) constrain all initial orbital elements except the true longitude
5) constrain semimajor axis, eccentricity and inclination; bounded RAAN and true longitude

For both options, the initial orbit inclination is constrained by enforcing

$$\sqrt{h^2 + k^2} = \tan\left(\frac{i}{2}\right)$$

where *i* is the user-defined park orbit inclination. Furthermore, the semi-parameter is constrained to the initial value according to $p_L = p_U = p_i$.

If the park orbit is circular, the software enforces the two constraints $f = 0$ and $g = 0$.

Otherwise, for an elliptical park orbit, the single equality constraint $\sqrt{f^2 + g^2} = e$ is enforced, where *e* is the park orbit eccentricity.

For program option 1, both lower and upper bounds for the *h* and *k* modified equinoctial elements are set equal to the initial elements as

$$h_L = h_U = h_i \qquad k_L = k_U = k_i$$

For both options, the initial true longitude is bounded according to $-360^\circ \le L_i \le +360^\circ$ and for option 2, the RAAN of the initial park orbit is bounded according to

$$\Omega_i - 30^\circ \le \Omega \le \Omega_i + 30^\circ$$

where Ω_i is the user's initial guess for RAAN.

The spacecraft mass at the initial time is constrained to the user-defined initial value.

In *SOS* control terminology, these constraints or boundary conditions are called *point functions*.

Performance index – maximize final spacecraft mass

The objective function or performance index *J* for this simulation is the mass of the spacecraft at burnout of the lunar injection stage. This is simply $J = m_f$.

The value of the `maxmin` indicator in the code tells the software whether the user is minimizing or maximizing the performance index. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass.

Path constraint – unit thrust vector scalar magnitude

During the TLI propulsive maneuver, the scalar magnitude of the components of the unit thrust vector at any time during the maneuver is constrained as follows:

$$\left|\mathbf{u}_T\right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

Point functions – periapsis altitude and selenocentric orbital inclination

At the user-defined sphere-of-influence of the Moon, the point function enforced by the software is given by $r_{SOI_p} - r_{SOI} = 0$ where r_{SOI_p} is the *predicted* SOI radius and r_{SOI} is the SOI radius defined by the user in the simulation definition file.

Targeting to a selenocentric periapsis radius and orbital inclination

For user-defined periapsis radius and orbital inclination targets at the moon, the following point functions or equality constraints are enforced.

$$r_p - r_{ca} = 0 \qquad \cos i - \hat{\mathbf{h}}_z = 0$$

In these equality constraints, r_p and *i* are the user-defined periapsis radius and selenocentric orbital inclination of the encounter hyperbola, respectively. In the second equation, $\hat{\mathbf{h}}_z$ is the z-component of the predicted unit angular momentum vector. These orbital elements are determined from the spacecraft's state vector at closest approach to the moon. The orbital inclination point function is expressed in the lunar mean equator and IAU node of epoch coordinate system which is described in *Appendix G – Aerospace Trajectory Coordinates and Time Systems.*

The elapsed time from the lunar SOI until closest approach to the moon is determined by an algorithm that includes Brent's one-dimensional root-finder embedded within a Runge-Kutta-Fehlberg 7(8) numerical integration method. This technique searches for the time at which the selenocentric flight path angle γ of the spacecraft is zero. This mission constraint is computed as

$$\gamma = \sin^{-1}\left(\frac{\mathbf{r}\bullet\mathbf{v}}{|\mathbf{r}\bullet\mathbf{v}|}\right) = 0$$

where **r** and **v** are the selenocentric position and velocity vectors, respectively.

Technical discussion

During Phase 1 – TLI propulsive maneuver, the spacecraft motion is modeled using modified equinoctial equations of motion. Please refer to *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for the geometry and dynamics in this system. During Phase 2 – coast to lunar closest approach, the spacecraft motion is modeled using cartesian equations of motion. *Appendix C – Cartesian Equations of Motion* describes this system along with perturbations due to Earth gravity, aerodynamic drag and other perturbations.

The spacecraft's orbital motion is modeled with respect to the Earth mean equator and equinox of J2000 (EME2000) coordinate system. Please refer to *Appendix G – Aerospace Trajectory Coordinates and Time Systems*.

Propulsive thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)}\hat{\mathbf{u}}_T(t)$$

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_r} & u_{T_t} & u_{T_n} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system.

The components of this unit vector are the control variables for this problem.

The propellant mass flow rate is determined from

$$\dot{m} = \frac{dm}{dt} = \frac{T}{g\, I_{sp}}$$

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g\, I_{sp}$ is also called the *exhaust velocity*. The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}\, t$ where m_{sc_i} is the initial mass of the spacecraft.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as follows

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

Finally, the pitch and yaw angles can be determined from the components of the unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

The pitch angle is positive above the "local horizontal" and the yaw angle is positive in the direction of the angular momentum vector.

In the `tlto_sos` computer program, the components of the inertial unit thrust vector are defined in terms of the right ascension α and the declination angle δ as follows:

$$u_{T_{ECI_x}} = \cos\alpha\cos\delta \qquad u_{T_{ECI_y}} = \sin\alpha\cos\delta \qquad u_{T_{ECI_z}} = \sin\delta$$

Finally, the right ascension and declination angles can be determined from the components of the ECI unit thrust vector according to

$$\alpha = \tan^{-1}\left(u_{T_{ECI_y}}, u_{T_{ECI_x}}\right) \qquad \delta = \sin^{-1}\left(u_{T_{ECI_z}}\right)$$

The B-plane

The geometry and fundamental equations of the B-plane system can be found in *Appendix D – B-Plane Geometry, Coordinates and Targeting*.

Geocentric-to-selenocentric coordinate transformation

Please consult *Appendix G – Aerospace Trajectory Coordinates and Time Systems* for the geometry and transformation equations related to this system.

Circularization delta-v

The impulsive delta-v required to circularize the spacecraft's trajectory at closest approach to the moon can be computed from

$$\Delta v = v_p - \sqrt{\frac{\mu_m}{r_p}} = v_p - v_{lc}$$

where v_p is the velocity of the incoming hyperbola at periapsis, r_p is the periapsis radius at closest approach, and μ_m is the gravitational constant of the moon. For capture into an elliptical orbit at the moon, the impulsive delta-v is determined using

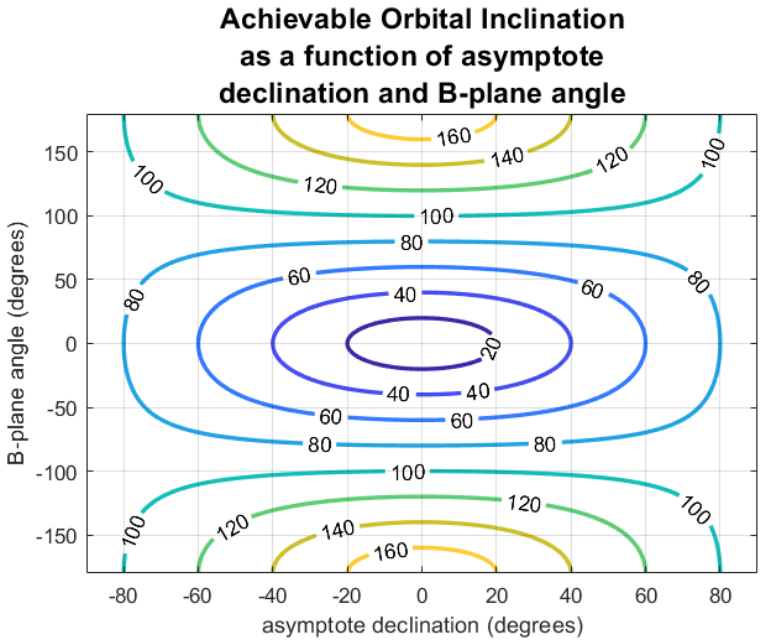
$$\Delta v = v_p - \sqrt{\frac{2\mu_m}{r_p} + \frac{\mu_m}{a}}$$

where *a* is the semimajor axis of the final elliptical orbit.

A note about targeting the lunar inclination

The range of orbital inclinations possible at closest approach to the moon is a function of the declination of the incoming hyperbola. This range is governed by the following constraint $i > |\delta_\infty|$ where *i* is the selenocentric inclination of the final lunar orbit and δ_∞ is the selenocentric declination of the incoming hyperbola.

The relationship between orbital inclination *i*, B-plane angle θ and asymptote declination δ of an incoming or outgoing hyperbola is given by $\cos i = \cos\theta \cos\delta$. The following is a contour plot illustrating the *achievable* inclination as a function of B-plane angle and declination.



Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and CSV data file produced by the `tlto_sos` software.

The simulation summary screen display contains the following information.

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees.

period (min) = orbital period.

rx (km) = x-component of the eci position vector in kilometers

ry (km) = y-component of the eci position vector in kilometers

rz (km) = z-component of the eci position vector in kilometers

rmag (km) = geocentric position magnitude in kilometers

vx (kps) = x-component of the eci velocity vector in kilometers/second

vy (kps) = y-component of the eci velocity vector in kilometers/second

vz (kps) = z-component of the eci velocity vector in kilometers/second

vmag (kps) = velocity vector scalar magnitude in kilometers/seconds

deltav = scalar magnitude of the TLI maneuver in meters/seconds

The user-defined comma-separated-variable disk file is created by the `odeprt` subroutine and contains the following information:

time (hrs) = time since ignition in hours

rx (km) = x-component of eci position vector in kilometers

ry (km) = y-component of eci position vector in kilometers

rz (km) = z-component of eci position vector in kilometers

rmag (km) = geocentric radius magnitude in kilometers

vx (km/sec) = x-component of eci velocity vector in kilometers per second

vy (km/sec) = y-component of eci velocity vector in kilometers per second

vz (km/sec) = z-component of eci velocity vector in kilometers per second

vmag (km/sec) = scalar velocity vector in kilometers per second

semimajor axis (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

arg of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

fpa (deg) = flight path angle in degrees

seleno radius (km) = selenocentric radius of the spacecraft in kilometers

rm2sc-x (km) = x-component of selenocentric position vector in kilometers

rm2sc-y (km) = y-component of selenocentric position vector in kilometers

rm2sc-z (km) = z-component of selenocentric position vector in kilometers

rm2scm (km) = selenocentric position magnitude in kilometers

pmee = orbital semiparameter in kilometers

fmee = modified equinoctial orbital element = ecc * cos(argper + raan)

gmee = modified equinoctial orbital element = ecc * sin(argper + raan)

hmee = modified equinoctial orbital element = tan(i/2) * cos(raan)

xkmee = modified equinoctial orbital element = tan(i/2) * sin(raan)

xlmee = modified equinoctial orbital element = orbital true longitude in degrees

rmoon-x (km) = x-component of the moon’s geocentric position vector in kilometers

rmoon-y (km) = y-component of the moon’s geocentric position vector in kilometers

rmoon-z (km) = z-component of the moon’s geocentric position vector in kilometers

yaw (deg) = thrust vector yaw angle in degrees

pitch (deg) = thrust vector pitch angle in degrees

rasc (deg) = thrust vector right ascension angle in degrees

decl (deg) = thrust vector declination angle in degrees

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

ut-eci-x = x-component of the eci unit thrust vector

ut-eci-y = y-component of the eci unit thrust vector

ut-eci-z = z-component of the eci unit thrust vector

"Lunar Trajectories", NASA TN D-866, August 1961.

"Earth-Moon Trajectories", JPL Technical Report No. 32-503, May 1, 1964.

"Three-Dimensional Lunar Trajectories", V. A. Egorov, Mechanics of Space Flight Series, Israel Program for Scientific Translations, Jerusalem 1969.

"Circumlunar Trajectory Calculations", MIT Instrumentation Laboratory Report R-353, April 1962.

"Optimal Low Thrust Trajectories to the Moon", John T. Betts and Sven O. Erb, *SIAM Journal on Applied Dynamical Systems*, Vol. 2, No. 2, pp. 144-170, 2003.

CAMTO 11 – Ballistic Interplanetary Trajectory Optimization

This *CMATO* application is a Fortran computer program named `ipto_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the classic one impulse flyby and two-impulse rendezvous ballistic interplanetary trajectory optimization problems. The software attempts to minimize the launch delta-v, the arrival delta-v or the total delta-v for the interplanetary transfer. The type of trajectory optimization is specified by the user. The destination celestial body can be a comet or asteroid.

The important features of this scientific simulation are as follows:

- one impulse, *patched-conic* interplanetary *flyby* trajectory modeling
- two impulse, *patched-conic* interplanetary *rendezvous* trajectory modeling
- n-body heliocentric, inertial cartesian equations of motion
- elliptical, non-coplanar asteroid and comet orbits
- JPL DE421 planetary ephemeris model

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `ipto_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's problem and perform the optimization using a sparse nonlinear programming (NLP) method.

The `ipto_sos` software allows the user to select the type of collocation method and other important algorithm control parameters.

Input file format and contents

The `ipto_sos` computer program is "data-driven" by a simple user-created text file. The following is a typical input or "simulation definition" file used by the software. This example is an Earth-to-Mars ballistic rendezvous trajectory that minimizes the total delta-v for the mission. Please note for a flyby trajectory input file (`trajectory type = 1`), the only valid optimization option is `1 = minimize launch delta-v`.

In the following discussion the actual input file contents are in *`courier`* font and all explanations are in times font. Please note that the fundamental time argument in this computer program is Barycentric Dynamical Time (TDB).

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is <u>not</u> case sensitive but must be spelled correctly.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*********************************************************
** impulsive delta-v interplanetary trajectory optimization
** patched-conic heliocentric motion - ipto_sos
** Mars '03 - mars03.in
** May 6, 2011
*********************************************************
```

The first program input is an integer that defines the type of delta-v optimization. Please note that option 4, `no optimization` simply solves the n-body, orbital two-point boundary value problem.

```
******************
* simulation type *
******************
 1 = minimize launch delta-v
 2 = minimize arrival delta-v
 3 = minimize total delta-v
 4 = no optimization
-----------------------------
3
```

The next input is an integer that tells the simulation what type of trajectory to model.

```
trajectory type (1 = flyby, 2 = rendezvous)
2
```

The next three inputs are the user's initial guess for the launch calendar date. Be sure to include all four digits of the calendar year.

```
launch calendar date initial guess (month, day, year)
6,1,2003
```

The software allows the user to specify an initial guess for the launch and arrival calendar dates and lower and upper bounds on the actual dates found during the optimization process. For any guess for

launch time t_L and user-defined launch time lower and upper bounds Δt_l and Δt_u, the launch time t is constrained as follows:

$$t_L - \Delta t_l \le t \le t_L + \Delta t_u$$

Likewise, for any guess for arrival time t_A and user-defined arrival time bounds Δt_l and Δt_u, the arrival time t is constrained as follows:

$$t_A - \Delta t_l \le t \le t_A + \Delta t_u$$

For fixed launch and/or arrival times, the lower and upper bounds should be set to 0.

The next two inputs are the lower and upper bounds for the launch calendar date search interval. These values should be input in days.

```
launch date search boundary (days)
-30, +30
```

The next program input is an integer that specifies the launch planet.

```
*****************
* launch planet *
*****************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
----------
3
```

The next set of inputs defines the user's initial guess for the arrival calendar date, the search interval and the arrival planet/comet/asteroid.

```
arrival calendar date initial guess (month, day, year)
12,1,2003

arrival date search boundary (days)
-30, +30

*************************
* arrival celestial body *
*************************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
 0 = asteroid/comet
-------------------
4
```

The next series of inputs include the name and classical orbital elements of a comet or asteroid (arrival celestial body = 0). Please note that the angular orbital elements must be specified with respect to a

heliocentric, Earth mean ecliptic and equinox of J2000 coordinate system. The calendar date of perihelion passage should be with respect to the TDB time system. Please consult Appendix F – Computing an Asteroid or Comet Ephemeris for additional information about this option.

```
*****************************************
* asteroid/comet classical orbital elements *
* (heliocentric, Earth mean ecliptic J2000) *
*****************************************

asteroid/comet name
Tempel 1

calendar date of perihelion passage (month, day, year)
7, 5.3153, 2005

perihelion distance (au)
1.506167

orbital eccentricity (non-dimensional)
0.517491

orbital inclination (degrees)
10.5301

argument of perihelion (degrees)
178.8390

longitude of the ascending node (degrees)
68.9734
```

This next integer input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by the *Sparse Optimization Suite*. Options 2 and 3 allow the user to specify either the number of nodes (option 2) or time step size of the data file (option 3).

```
*****************************************
* type of comma-delimited solution data file *
*****************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this next input defines either the number of data points (option 2) or the time step size of the data output in the solution file (option 3).

```
number of user-defined nodes or print step size in solution data file
1
```

The name of the solution data file is defined in this next line. Information about the contents of this file is provided later in this document.

```
name of solution output file
mars03.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
* algorithm control parameters *
```

```
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the *Sparse Optimization Suite* optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the differential equation algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
**************************
differential equation output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

Optimal control solution

The following is the program output created by the `ipto_sos` simulation for this example. The first part of the solution display summarizes important information about the type of trajectory and the type of optimization and the launch conditions. Please note the launch hyperbola coordinates are with respect to the Earth mean equator and equinox of J2000 (EME2000) system. The second part of the display summarizes the arrival conditions and characteristics of the planetary and heliocentric transfer orbits.

```
 program ipto_sos
 ================

 rendezvous trajectory
 minimize total delta-v

 DE421 ephemeris

 DEPARTURE CONDITIONS
 ====================

 calendar date       06/06/2003

 TDB time            08:18:17.36

 TDB julian date     2452796.84603420

 heliocentric delta-v vector and magnitude
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

 launch delta-vx     2900.35549073551      meters/second
 launch delta-vy    -616.714043478893      meters/second
 launch delta-vz    -40.1696083298423      meters/second

 launch delta-v      2965.46990905271      meters/second

 launch hyperbola characteristics
 (Earth mean equator and equinox of J2000)
 -----------------------------------------

 right ascension     349.265305839434      degrees
```

```
declination        -5.46005203692079      degrees

C3                  8.79401178149709      (km/sec)**2

heliocentric orbital elements and state vector of the departure planet
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)         eccentricity       inclination (deg)     argper (deg)
 0.1000231776D+01    0.1633787580D-01    0.3276160283D-03    0.2719030375D+03

    raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.1902549286D+03    0.1530415819D+03    0.6494461938D+02    0.3653838922D+03

     rx (km)             ry (km)             rz (km)            rmag (km)
 -.3877865233D+08    -.1467666196D+09    0.7863271593D+03    0.1518032427D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 0.2832127123D+02    -.7711221751D+01    0.7221819658D-04    0.2935229710D+02

heliocentric orbital elements and state vector of the spacecraft
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)         eccentricity       inclination (deg)     argper (deg)
 0.1259528064D+01    0.1943611232D+00    0.7109964954D-01    0.1789331987D+03
    raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.7543871484D+02    0.8276342515D+00    0.1797608329D+03    0.5163095994D+03

     rx (km)             ry (km)             rz (km)            rmag (km)
 -.3877865235D+08    -.1467666197D+09    0.7863271176D+03    0.1518032428D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 0.3122162673D+02    -.8327935797D+01    -.4009739015D-01    0.3231324958D+02
```

ARRIVAL CONDITIONS

```
==================

calendar date       12/27/2003

TDB time            16:57:09.05

TDB julian date     2453001.20635469

heliocentric delta-v vector and magnitude
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

arrival delta-vx    2021.79887865832      meters/second
arrival delta-vy   -1614.33782179448      meters/second
arrival delta-vz   -779.406642586644      meters/second

arrival delta-v     2702.07920371116      meters/second

C3                  7.30123202312836      (km/sec)**2

heliocentric orbital elements and state vector of the planet/asteroid/comet
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)         eccentricity       inclination (deg)     argper (deg)
 0.1523678538D+01    0.9354080419D-01    0.1849372004D+01    0.2865170844D+03

    raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.4954092332D+02    0.7248490912D+02    0.3590019936D+03    0.6869710761D+03

     rx (km)             ry (km)             rz (km)            rmag (km)
 0.1454912476D+09    0.1646993683D+09    -.1235263971D+06    0.2197580494D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 -.1723361516D+02    0.1810898471D+02    0.8028150130D+00    0.2501154394D+02

heliocentric orbital elements and state vector of the spacecraft
(Earth mean ecliptic and equinox of J2000)
```

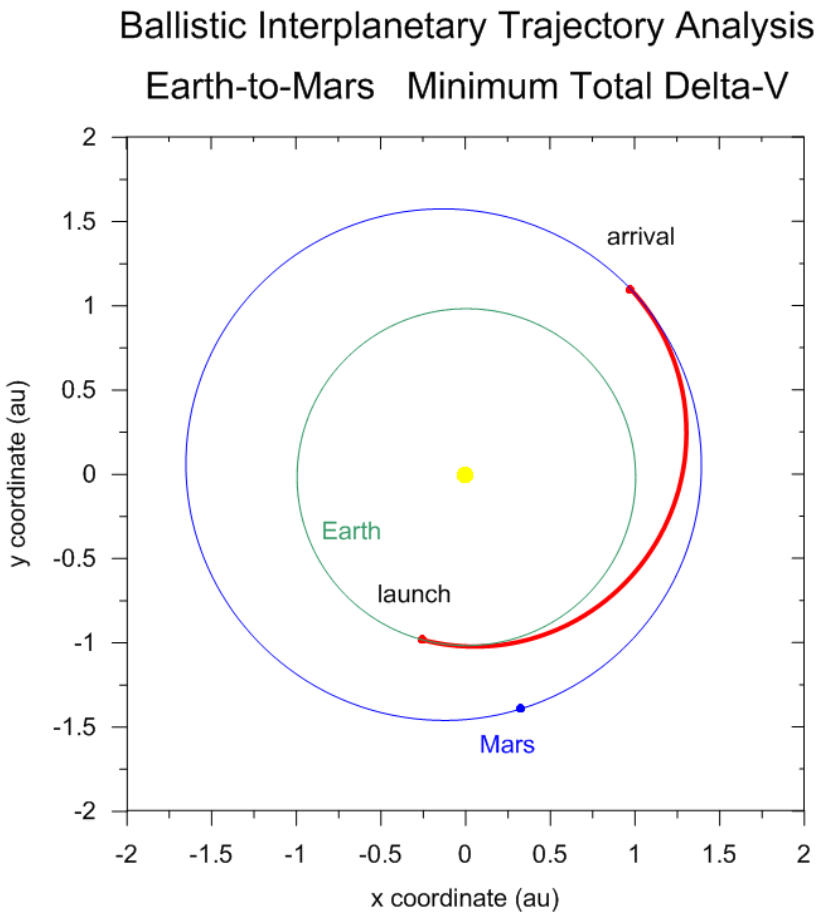
```
-----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.1523678539D+01    0.9354080441D-01    0.1849372004D+01    0.2865170849D+03

    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.4954092332D+02    0.7248490871D+02    0.3590019936D+03    0.6869710771D+03

     rx (km)             ry (km)             rz (km)             rmag (km)
 0.1454912476D+09    0.1646993683D+09    -.1235263972D+06    0.2197580495D+09

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 -.1723361516D+02    0.1810898471D+02    0.8028150132D+00    0.2501154395D+02
```

The simulation also provides the total delta-v for the mission along with the transfer time in days.

```
total delta-v       5667.54911276387      meters/second

transfer time       204.360320485197      days
```

The `ipto_sos` software will create two comma-separated-variable (csv) output files. The first file contains the heliocentric, ecliptic state vector of the spacecraft and the second file (`planets.csv`) contains the state vectors of the planet and the destination celestial body. These data files can be used to create graphic displays of the trajectory.

The following is the transfer trajectory for this example. It is a view of the trajectory and planetary orbits from the north pole of the ecliptic looking down on the ecliptic plane. The Earth's orbit trace is green and the orbit of Mars is blue. The transfer trajectory is red and the x-y scales are in astronomical units. The small blue dot is the location of Mars at Earth departure.



Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* can be verified by numerically integrating the n-body orbital equations of motion with the *SOS*-computed initial departure conditions and the optimal delta-v vectors. This is equivalent to solving an initial value problem (IVP) that uses the optimal solution. This part of the `ipto_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is a display of the final solution computed using this *explicit* numerical integration method. This display includes the final heliocentric orbital elements of the spacecraft along with the position and velocity "matching" errors.

```
=======================================
verification of optimal control solution
=======================================

heliocentric orbital elements and state vector of the spacecraft at arrival
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.1523678462D+01    0.9354076752D-01    0.1849372014D+01    0.2865170728D+03

    raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.4954092238D+02    0.7248492066D+02    0.3590019934D+03    0.6869710250D+03

     rx (km)             ry (km)             rz (km)            rmag (km)
 0.1454912486D+09    0.1646993631D+09    -.1235264148D+06    0.2197580462D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 -.1723361512D+02    0.1810898443D+02    0.8028150120D+00    0.2501154372D+02

final heliocentric position vector and magnitude errors

delta rx            1.04274883866310      kilometers
delta ry           -5.17958366870880      kilometers
delta rz          -1.770855819631834E-002 kilometers

delta rmag          5.28353344975894      kilometers

final heliocentric velocity vector and magnitude errors

delta vx           3.511720336746293E-008 kilometers/second
delta vy          -2.801622756010147E-007 kilometers/second
delta vz          -9.508036580285761E-010 kilometers/second

delta vmag         2.823561982140410E-007 kilometers/second
```

Creating an initial guess

An initial guess for the launch and arrival impulsive delta-v vectors can be determined from the solution of the Lambert two-point boundary-value problem (TPBVP). Lambert's Theorem states that the time to traverse a trajectory depends only upon the length of the semimajor axis *a* of the transfer trajectory, the sum $r_i + r_f$ of the distances of the initial and final positions relative to a central body, and the length *c* of the chord joining these two positions. Please consult Appendix I – Numerical Solutions of Lambert's Problem for additional information about Lambert's problem.

The Lambert solution that initializes the `ipto_sos` software uses the user's initial guess for launch and arrival dates. The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine for this aerospace trajectory optimization problem. These program options create an initial guess from the numerical

integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 2` program option tells the software to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

The algorithm used in this computer program to solve the two-body Lambert problem is based on the method described in "A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem" by R. H. Gooding, *Celestial Mechanics and Dynamical Astronomy* **48:** 145-165, 1990. This iterative solution is valid for elliptic, parabolic and hyperbolic transfer orbits which may be either posigrade or retrograde, and involve one or more revolutions about the central body.

Problem setup

This section provides additional details about the software implementation. For good scaling during the optimization, the time unit used in all internal calculations is days, position is expressed in astronomical units, and the unit for velocity and delta-v is astronomical units per day.

Launch and arrival time bounds

The software allows the user to specify an initial guess for the launch and arrival calendar dates and bounds on the actual dates found during the optimization process. For any guess for launch time t_L and user-defined launch time search bound Δt_L, the launch time t is constrained as follows:

$$t_L - \Delta t_L \le t \le t_L + \Delta t_L$$

Likewise, for any guess for arrival time t_A and user-defined arrival time bound Δt_A, the arrival time t is constrained as follows:

$$t_A - \Delta t_A \le t \le t_A + \Delta t_A$$

For fixed launch and/or arrival times, the lower and upper bounds are set to 0.

Performance index – minimize delta-v

The objective function or performance index *J* for this simulation is one of three possible delta-v's. For this classic trajectory optimization problem, this index is simply $J = \Delta V$ where ΔV is either the launch delta-v, arrival delta-v or the total delta-v. The value of the `maxmin` indicator in the *Sparse Optimization Suite* software tells the program whether the user is minimizing or maximizing the performance index. The type of delta-v optimization is selected by the user.

Point functions – position and velocity vector "matching" at launch

For any launch time t_L, the optimal solution for a rendezvous trajectory must satisfy the following state vector boundary conditions (equality constraints) at launch:

$$\mathbf{r}_{s/c}\left(t_L\right) - \mathbf{r}_p\left(t_L\right) = 0$$

$$\mathbf{V}_{s/c}\left(t_L\right) - \left\{\mathbf{V}_p\left(t_L\right) + \Delta\mathbf{V}\left(t_L\right)\right\} = 0$$

where $\mathbf{r}_{s/c}$ and $\mathbf{V}_{s/c}$ are the heliocentric, inertial position and velocity vectors of the spacecraft at the launch time t_L, $\mathbf{r}_p$ and $\mathbf{V}_p$ are the heliocentric, inertial position and velocity vectors of the departure planet at the launch time, and $\Delta\mathbf{V}$ is the *impulsive* heliocentric delta-v vector required at launch.

Point functions – position and/or velocity vector "matching" at arrival

For any arrival time t_A, the optimal solution for a rendezvous trajectory must satisfy the following state vector boundary conditions at arrival:

$$\mathbf{r}_{s/c}\left(t_A\right) - \mathbf{r}_p\left(t_A\right) = 0$$

$$\mathbf{V}_{s/c}\left(t_A\right) - \left\{\mathbf{V}_p\left(t_A\right) + \Delta\mathbf{V}\left(t_A\right)\right\} = 0$$

where $\mathbf{r}_{s/c}$ and $\mathbf{V}_{s/c}$ are the heliocentric, inertial position and velocity vectors of the spacecraft at the arrival time t_A, and $\mathbf{r}_p$ and $\mathbf{V}_p$ are the heliocentric, inertial position and velocity vectors of the destination planet at the arrival time, and ΔV is the *impulsive* heliocentric delta-v vector required at arrival.

This system of launch and arrival state vector equality constraints ensures a rendezvous mission. For a flyby mission, the velocity vector point functions at arrival are not enforced.

The components of the departure and arrival impulsive delta-v's are the optimization parameters used by the Sparse Optimization Suite to solve this trajectory problem.

Technical discussion

The spacecraft motion is modeled using Cowell's form of the heliocentric equations of motion. Please consult Appendix C – Cartesian Equations of Motion for the numerical methods implemented in this software.

The deltav's required at launch and arrival are simply the differences between the velocity on the optimal transfer trajectory and the heliocentric velocities of the two celestial objects. If we treat each planet as a point mass and assume *impulsive* maneuvers, the *planet-centered* magnitude and direction of the required maneuvers are given by the two vector equations:

$$\Delta\mathbf{V}_L = \mathbf{V}_{T_L} - \mathbf{V}_{P_L}$$

$$\Delta\mathbf{V}_A = \mathbf{V}_{P_A} - \mathbf{V}_{T_A}$$

where

$\mathbf{V}_{T_L}$ = heliocentric velocity vector of the transfer trajectory at launch

$\mathbf{V}_{T_A}$ = heliocentric velocity vector of the transfer trajectory at arrival

$\mathbf{V}_{P_L}$ = heliocentric velocity vector of the launch planet

$\mathbf{V}_{P_A}$ = heliocentric velocity vector of the arrival planet

The scalar magnitude of each maneuver is also called the "hyperbolic excess velocity" or V_∞ at launch and arrival. The hyperbolic excess velocity is the speed of the spacecraft relative to each planet or celestial body at an *infinite* distance from the planet. Furthermore, the *energy* or C_3 at launch or arrival is equal to V_∞^2 for the respective maneuver. C_3 is also equal to twice the orbital energy per unit mass (the specific orbital energy).

The orientation of the departure hyperbola is specified in terms of the right ascension and declination of the outgoing asymptote. These coordinates can be calculated using the Cartesian components of the V_∞ velocity vector.

The right ascension of the asymptote is determined from $\alpha = \tan^{-1}\left(\Delta V_y, \Delta V_z\right)$ and the geocentric declination of the asymptote is given by $\delta = 90^0 - \cos^{-1}\left(\Delta \hat{V}_z\right)$ where $\Delta \hat{V}_x$, $\Delta \hat{V}_y$ and $\Delta \hat{V}_z$ are the x, y and z components of the unit ΔV vector. The right ascension is computed using a four-quadrant inverse tangent function.

In this computer program the heliocentric planetary coordinates and therefore the ΔV vectors are computed in the Earth mean ecliptic and equinox of J2000 coordinate system. This coordinate system along with useful transformations can be found in Appendix G – Aerospace Trajectory Coordinates and Time Systems

Earth-to-Tempel 1 flyby analysis

This section summarizes typical trajectory characteristics of a ballistic flyby mission from Earth to the comet Tempel 1. This simulation minimizes the magnitude of the launch delta-v at the Earth departure. Please note for a flyby trajectory input file (`trajectory type = 1`), the only valid delta-v optimization option is `1 = minimize launch delta-v`.

Here is the `ipto_sos` input data file for this example.

```
************************************************************
** impulsive delta-v interplanetary trajectory optimization
** ballistic patched-conic heliocentric motion - ipto_sos
** Earth-to-Tempel 1 flyby - tempel1.in
** May 6, 2011
************************************************************

******************
* simulation type *
******************
 1 = minimize launch delta-v
 2 = minimize arrival delta-v
 3 = minimize total delta-v
 4 = no optimization
-----------------------------
1

trajectory type (1 = flyby, 2 = rendezvous)
1

launch calendar date initial guess (month, day, year)
12,1,2004

launch date search boundary (days)
-60, +60
```

```
*****************
* launch planet *
*****************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
----------
3

arrival calendar date initial guess (month, day, year)
7,1,2005

arrival date search boundary (days)
-90, +90

*************************
* arrival celestial body *
*************************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
 0 = asteroid/comet
--------------------
0

*******************************************
* asteroid/comet classical orbital elements *
* (heliocentric, Earth mean ecliptic J2000) *
*******************************************

asteroid/comet name
Tempel 1

calendar date of perihelion passage (month, day, year)
7, 5.3153, 2005

perihelion distance (au)
1.506167

orbital eccentricity (non-dimensional)
0.517491

orbital inclination (degrees)
10.5301

argument of perihelion (degrees)
178.8390

longitude of the ascending node (degrees)
68.9734

*********************************************
* type of comma-delimited solution data file *
*********************************************
 1 = OC-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1

number of user-defined nodes or print step size in solution data file
1
```

```
name of solution output file
tempel1.csv

*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
---------------------------------
1

relative error in the objective function (performance index)
1.0d-5

relative error in the solution of the differential equations
1.0d-7

maximum number of mesh refinement iterations
20

maximum number of function evaluations
50000

maximum number of algorithm iterations
5000

**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1

**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1

**************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1

******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

Here's the optimal solution for this example including the verification results.

```
 program ipto_sos
 ================
```

flyby trajectory

minimize launch delta-v

DE421 ephemeris

DEPARTURE CONDITIONS
==========

calendar date 01/10/2005

TDB time 08:45:26.84

TDB julian date 2453380.86489403

heliocentric delta-v vector and magnitude
(Earth mean ecliptic and equinox of J2000)
---

launch delta-vx -2971.11698008735 meters/second
launch delta-vy -1191.71198647956 meters/second
launch delta-vz -335.134462946394 meters/second

launch delta-v 3218.69984253530 meters/second

launch hyperbola characteristics
(Earth mean equator and equinox of J2000)
---

right ascension 197.907324746402 degrees

declination -14.0521359432754 degrees

C3 10.3600286763368 (km/sec)**2

heliocentric orbital elements and state vector of the departure planet
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1000965820D+01	0.1761052573D-01	0.7749154801D-03	0.3176116281D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.1455924113D+03	0.6945172026D+01	0.3245568001D+03	0.3657861848D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.5067923758D+08	0.1381198216D+09	-.1153891645D+04	0.1471239962D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.2846316247D+02	-.1037623019D+02	0.3333146047D-03	0.3029550743D+02

heliocentric orbital elements and state vector of the spacecraft
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1300685590D+01	0.2438903757D+00	0.5726808349D+00	0.1803234359D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.2901042542D+03	0.3597215236D+03	0.1800449595D+03	0.5418223555D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.5067923759D+08	0.1381198216D+09	-.1153891645D+04	0.1471239962D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.3143427945D+02	-.1156794217D+02	-.3348011483D+00	0.3349691482D+02

ARRIVAL CONDITIONS
==========

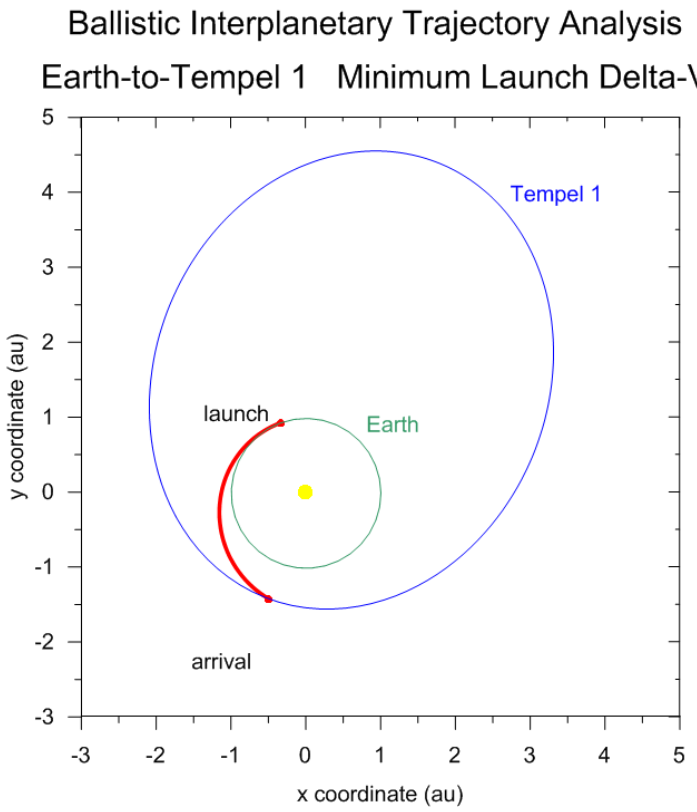
calendar date 07/10/2005

TDB time 02:21:44.73

TDB julian date 2453561.59843438

```
 heliocentric orbital elements and state vector of the planet/asteroid/comet
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

      sma (au)          eccentricity       inclination (deg)     argper (deg)
  0.3121531412D+01    0.5174910000D+00    0.1053010000D+02    0.1788390000D+03

     raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
  0.6897340000D+02    0.3140665417D+01    0.1819796654D+03    0.2014419845D+04

      rx (km)             ry (km)             rz (km)             rmag (km)
  -.7369140595D+08    -.2130455810D+09    -.1423200348D+07    0.2254348429D+09

      vx (kps)            vy (kps)            vz (kps)            vmag (kps)
  0.2759316336D+02    -.1009890910D+02    -.5461105864D+01    0.2988635652D+02

heliocentric orbital elements and state vector of the spacecraft
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

      sma (au)          eccentricity       inclination (deg)     argper (deg)
  0.3501454000D+01    0.5700007349D+00    0.2868171372D+01    0.1832730028D+03

     raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
  0.6368061244D+02    0.3975119825D+01    0.1872481226D+03    0.2393156305D+04

      rx (km)             ry (km)             rz (km)             rmag (km)
  -.7369140595D+08    -.2130455810D+09    -.1423200348D+07    0.2254348429D+09

      vx (kps)            vy (kps)            vz (kps)            vmag (kps)
  0.2843623711D+02    -.1063584982D+02    -.1513252337D+01    0.3039787517D+02

 total delta-v       3218.69984253530      meters/second

 transfer time       180.733540346846      days

 ========================================
 verification of optimal control solution
 ========================================

 heliocentric orbital elements and state vector of the spacecraft at arrival
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

      sma (au)          eccentricity       inclination (deg)     argper (deg)
  0.3501453652D+01    0.5700006990D+00    0.2868171382D+01    0.1832730062D+03

     raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
  0.6368061198D+02    0.3975116501D+01    0.1872481227D+03    0.2393155948D+04

      rx (km)             ry (km)             rz (km)             rmag (km)
  -.7369140625D+08    -.2130455767D+09    -.1423200336D+07    0.2254348390D+09

      vx (kps)            vy (kps)            vz (kps)            vmag (kps)
  0.2843623719D+02    -.1063584937D+02    -.1513252335D+01    0.3039787509D+02

 final heliocentric position vector and magnitude errors

 delta rx          -0.301263228058815      kilometers
 delta ry            4.24355044960976      kilometers
 delta rz           1.195031986571848E-002 kilometers

 delta rmag          4.25424761398634      kilometers
```

Here's the graphics display of the interplanetary transfer trajectory along with the planet and comet heliocentric orbits. The transfer trajectory is red and the scales are in astronomical units. For this example we can see that encounter with Tempel 1 occurs near perihelion of the comet's orbit.



Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `ipto_sos` software. All output except the coordinates of the launch hyperbola is computed and displayed in a heliocentric, Earth mean ecliptic and equinox of J2000 coordinate system.

The simulation summary screen display contains the following information:

calendar date = calendar date of trajectory event

ephemeris time = ephemeris time of trajectory event

julian date = julian date of trajectory event

right ascension = right ascension of the launch hyperbola in degrees

declination = declination of the launch hyperbola in degrees

sma (au) = semimajor axis in astronomical unit

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (days) = orbital period in days

delta-vx = x-component of the impulsive velocity vector in kilometers/second

delta-vy = y-component of the impulsive velocity vector in kilometers/second

delta-vz = z-component of the impulsive velocity vector in kilometers/second

delta-v = scalar magnitude of the impulsive delta-v in kilometers/seconds

energy (km/sec) = twice specific orbital energy in km(sec)**2

The comma-separated-variable disk file is created by the `odeprt` subroutine and contains the following information:

time (days) = simulation time since launch in days

rx (au) = x-component of spacecraft position vector in astronomical units

ry (au) = y-component of spacecraft position vector in astronomical units

rz (au) = z-component of spacecraft position vector in astronomical units

rmag (au) = heliocentric radius magnitude of spacecraft in astronomical units

vx (km/sec) = x-component of spacecraft velocity vector in kilometers per second

vy (km/sec) = y-component of spacecraft velocity vector in kilometers per second

vz (km/sec) = z-component of spacecraft velocity vector in kilometers per second

vmag (km/sec) = heliocentric velocity of spacecraft in kilometers per second

semimajor axis (au) = semimajor axis in astronomical units

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

arg of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (days) = orbital period in days

The `planets.csv` file contains the following information:

time (days) = simulation time since launch in days

rp1-x (au) = x-component of the launch planet position vector in astronomical units
rp1-y (au) = y-component of the launch planet position vector in astronomical units
rp1-z (au) = z-component of the launch planet position vector in astronomical units

rp2-x (au) = x-component of the destination body position vector in astronomical units
rp2-y (au) = y-component of the destination body position vector in astronomical units
rp2-z (au) = z-component of the destination body position vector in astronomical units

"Update to Mars Coordinate Frame Definitions", R. A. Mase, JPL IOM 312.B/015-99, 15 July 1999.

"The Planetary and Lunar Ephemeris DE 421", W. M. Folkner, J. G. Williams, and D. H. Boggs, JPL IOM 343R-08-003, 31 March 2008.

"Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000", *Celestial Mechanics and Dynamical Astronomy*, **82**: 83-110, 2002.

"IERS Conventions (2003)", IERS Technical Note 32, November 2003.

"Planetary Constants and Models", R. Vaughan, JPL D-12947, December 1995.

"Preliminary Mars Planetary Constants and Models for Mars Sample Return", D. Lyons, JPL IOM 312/99.DTL-1, 20 January 1999.

"Interplanetary Mission Design Handbook, Volume 1, Part 2", JPL Publication 82-43, September 15, 1983.

"An *Introduction to the Mathematics and Methods of Astrodynamics*, Revised Edition", Richard H. Battin, AIAA Education Series, 1999.

"A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem", Robert H. Gooding, *Celestial Mechanics and Dynamical Astronomy* **48:** 145-165, 1990.

CMATO 12 – Earth-to-Mars End-to-End Mission Design

This *CMATO* application is a Fortran computer program named e2m_ftn that can be used to design and optimize ballistic interplanetary missions from Earth park orbit to B-plane encounter at Mars. The software assumes that interplanetary injection occurs *impulsively* from a circular Earth park orbit. The B-plane coordinates are expressed in a Mars-centered (areocentric) mean equator and IAU node of epoch coordinate system. The B-plane targets are enforced using either a combination of flight path angle, periapsis radius and orbital inclination at a Mars-centered entry interface (EI) or individual B-plane coordinates ($\mathbf{B \cdot T}$ and $\mathbf{B \cdot R}$) of the arrival hyperbola. The type of targeting and the target values are defined by the user.

The software includes two options for creating an initial guess for the simulation. The first option allows the user to input the departure calendar date and time along with the C3, RLA and DLA "targets" for the Earth departure hyperbola. The second initial guess option computes these initial target conditions using an optimized two-body Lambert or "patched-conic" solution.

When the two-body Lambert initial guess option is selected, the first part of the e2m_ftn computer program solves for the minimum delta-v using a patched-conic, two-body Lambert solution for the transfer trajectory from Earth to Mars. The second part implements a simple *shooting* method that attempts to optimize the characteristics of the geocentric injection hyperbola while numerically integrating the spacecraft's geocentric and heliocentric equations of motion and targeting to components of the B-plane relative to Mars.

The spacecraft motion within the Earth's sphere-of-influence (SOI) includes non-spherical Earth gravity, the point-mass perturbation of the sun and moon, and optionally the point-mass perturbations of planets. The heliocentric equations of motion include the point-mass gravity of the sun, moon, and all planets of the solar system. The software also includes the option to include the effect of solar radiation pressure in both the geocentric and heliocentric phases of the spacecraft's orbital motion.

For the patched-conic initial guess option, the user can select one of the following delta-v optimization options for the two-body solution of the interplanetary transfer trajectory:

- minimize departure delta-v
- minimize arrival delta-v
- minimize total delta-v

The major computational steps implemented in this software are as follows:

- solve the two-body, patched-conic interplanetary Lambert problem for the energy C_3, declination (DLA) and asymptote (RLA) of the outgoing or departure hyperbola (only performed for the two-body Lambert initial guess)

- compute the orbital elements of the geocentric departure hyperbola and the components of the interplanetary injection delta-v vector

- perform geocentric orbit propagation from perigee of the geocentric departure hyperbola to the user-defined value of the Earth's sphere-of-influence (SOI)

- perform an n-body heliocentric orbit propagation from the Earth's SOI to either closest approach or entry interface (EI) at Mars

- target to the user-defined B-plane coordinates while minimizing the magnitude of the hyperbolic v-infinity at Earth departure (equivalent to minimizing the departure energy since $C_3 = V_\infty^2$)

This computer program uses a nonlinear programming (NLP) algorithm included in the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve both the patched-conic and numerically integrated trajectory optimization problems. The lunar, solar and planetary coordinates required by the software are computed using the JPL DE421 ephemeris.

Input file format and contents

The `e2m_ftn` software is "data-driven" by a user-created text file. The following is a typical input file used by this computer program. Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or change the number of lines reserved for each comment and data item. However, you may change them to reflect your own explanation or information. The annotation line also includes the correct units and when appropriate, the valid range of the input data items. ASCII text input is not case sensitive but must be spelled correctly. In the following discussion, the actual input file contents are in **`bold courier`** font and all explanations are in times font.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*****************************
** Earth-to-Mars interplanetary
** trajectory optimization
** Mars '03 example - mars03.in
** August 21, 2012
*****************************
```

The first program input is the name of a "constants and models" data file. This ASCII data file contains user-defined astrodynamic constants and other information.

```
name of constants and models data file
--------------------------------------
e2m_cm.dat
```

The following is a typical user-defined constants and models data file named `e2m_cm.dat`. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment or data item. Also, please note the proper units for each data item.

```
************************************
* e2m_ftn constants and models data file
************************************

astronomical unit (kilometers)
------------------------------
149597870.691d0

speed of light (meters/second)
------------------------------
299792458.0d0
```

```
solar flux at 1 AU (watts/meters**2)
------------------------------------
1366.1d0

Earth gravitational constant (km**3/sec**2)
-------------------------------------------
398600.4415d0

Earth equatorial radius (kilometers)
------------------------------------
6378.14d0

Earth sphere-of-influence value (kilometers)
--------------------------------------------
925000.0d0

Moon gravitational constant (km**3/sec**2)
------------------------------------------
4902.800238d0

Mars gravitational constant (km**3/sec**2)
------------------------------------------
42828.376212d0

Mars equatorial radius (kilometers)
-----------------------------------
3396.2d0
```

The second program input is the difference between ephemeris time (Terrestrial Time) and Universal Coordinated Time (UTC) in seconds.

```
ET-UTC (seconds)
----------------
64.132d0
```

The next program input is an integer that specifies the type of initial guess to use.

```
*********************
type of initial guess
*********************
 1 = user-defined
 2 = two-body Lambert
---------------------
1
```

The next series of data items are the user-defined values when using initial guess option 1. Please note that the right ascension (RLA) and declination (DLA) of the departure hyperbola must be input relative to the Earth mean equator and equinox of J2000 (EME2000) coordinate system.

```
*************************
user-defined initial guess
*************************

user-defined departure calendar date (month, day, year)
6, 5, 2003

user-defined departure UTC (hours, minutes, seconds)
14, 46, 20

user-defined C3 (kilometers^2/second^2)
8.787

user-defined EME2000 RLA (degrees; 0 <= RLA <= 360)
349.621
```

```
user-defined EME2000 DLA (degrees; -90 <= DLA <= +90)
-6.697
```

The next set of data items are used whenever the user has selected the two-body Lambert initial guess program option.

This data item is an integer that defines the type of patched-conic trajectory optimization.

```
*******************************
two-body Lambert simulation type
*******************************
 1 = minimize departure delta-v
 2 = minimize arrival delta-v
 3 = minimize total delta-v
---------------------------
1
```

The software allows the user to specify an initial guess for the departure and arrival calendar dates and a search interval. For any guess for departure time t_L and user-defined search interval Δt, the departure time t is constrained as follows

$$t_L - \Delta t \le t \le t_L + \Delta t$$

Likewise, for any guess for arrival time t_A and user-defined search interval, the arrival time t is constrained as follows

$$t_A - \Delta t \le t \le t_A + \Delta t$$

For fixed departure and/or arrival times, the search interval should be set to 0.

The next input defines an initial guess for the departure calendar date. Please be sure to include all digits of the calendar year.

```
departure calendar date initial guess (month, day, year)
6,1,2003
```

These two numbers define the lower and upper search interval for the departure calendar date.

```
departure date search boundary (days)
-30, +30
```

The next input defines an initial guess for the arrival calendar date.

```
arrival calendar date initial guess (month, day, year)
12,1,2003
```

These two numbers define the lower and upper search interval for the arrival calendar date.

```
arrival date search boundary (days)
-30, +30
```

The next two inputs define the altitude and orbital inclination of the circular park orbit.

```
************************
geocentric phase modeling
************************

park orbit altitude (kilometers)
185.32d0
```

```
Park orbit inclination (degrees)
28.5d0
```

The name of the ASCII data file containing the Earth gravity model data is specified in the next line. Please see the Technical Discussion section later in this document for a description and format of the data in this file.

```
name of Earth gravity model data file
egm96.dat
```

The order (zonals) of the Earth gravity model is an integer defined in the next line.

```
order of the gravity model (zonals)
8
```

The degree (tesserals) of the Earth gravity model is an integer defined in this next line.

```
degree of the gravity model (tesserals)
8
```

The next integer input "toggles" the option to include planetary point-mass perturbations during the geocentric phase of the targeting process.

```
include planetary point-mass perturbations (1 = yes, 0 = no)
1
```

This next integer specifies which injection opportunity is used by the software when calculating the characteristics of the departure hyperbola.

```
injection opportunity (1 or 2)
1
```

The next input specifies the type of targeting at Mars performed by the `e2m_ftn` computer program. Option 1 will target to user-defined components of the B-plane at closest approach to Mars, and option 2 will target to a Mars-centered hyperbola with a user-specified flight path angle, areocentric radius, and orbital inclination.

```
************************
Mars encounter targeting
************************

type of targeting
(1 = B-plane, 2 = orbital elements)
2
```

The next two inputs are the user-defined B-plane components used with targeting option 1.

```
B dot T
10965.19767

B dot R
-6109.045096
```

The next three inputs define the flight path angle, radius, and the orbital inclination of the encounter hyperbola at Mars. These flight conditions are used by targeting option 2. The radius is with respect to a spherical Mars model and the orbital inclination is with respect to the mean equator of Mars.

```
areocentric flight path angle (degrees)
0.0
```

```
areocentric radius (kilometers)
5000.0

areocentric orbital inclination (degrees)
60.0
```

The next series of inputs define the spacecraft characteristics used for solar radiation pressure perturbation calculations. These three items include the spacecraft's mass, reference cross-sectional area, and reflectivity coefficient. To exclude this perturbation, input a spacecraft mass of zero.

```
*********************************
spacecraft mass and SRP properties
*********************************

spacecraft mass (kilograms; input 0 to ignore SRP calculations)
3850.0d0

SRP reference area (square meters)
19.4

reflectivity coefficient (non-dimensional)
1.0d0
```

The last three inputs are algorithm control parameters that define the lower and upper bounds on the v-infinity, right ascension, and declination of the geocentric departure hyperbola.

```
***************************
algorithm control parameters
***************************

v-infinity lower and upper bounds increment (meters/second)
50.0

asymptote right ascension lower and upper bounds increment (degrees)
10.0

asymptote declination lower and upper bounds increment (degrees)
1.0
```

Program example

The following is the solution created with this computer program for this example. The output is organized by the following major sections

- *two-body/patched-conic pass*
    1) two body Lambert solution (if initial guess option 2)
    2) departure hyperbola orbital elements and state vector
    3) heliocentric coordinates of Earth at departure and Mars at arrival
    4) heliocentric coordinates of the spacecraft on the transfer trajectory

- *targeting/optimization pass*
    1) optimized characteristics of the departure hyperbola
    2) heliocentric coordinates of the spacecraft and Mars at encounter
    3) geocentric and heliocentric coordinates of the spacecraft at the Earth SOI

The first output section summarizes the two-body Lambert solution. The solution is provided in the heliocentric, Earth mean equator and equinox of J2000 (EME2000) coordinate system.

```
*****************************
two-body Lambert initial guess
*****************************

Earth-to-Mars mission design

=========================
two-body Lambert solution
=========================

minimize launch delta-v

departure heliocentric delta-v vector and magnitude
(Earth mean equator and equinox of J2000)
----------------------------------------

x-component of delta-v     2895.91191273315       meters/second
y-component of delta-v    -530.401772123313       meters/second
z-component of delta-v    -345.700686652980       meters/second

delta-v magnitude          2964.31118658849       meters/second

arrival heliocentric delta-v vector and magnitude
(Earth mean equator and equinox of J2000)
----------------------------------------

x-component of delta-v    -2063.01128433645       meters/second
y-component of delta-v     1164.27006011528       meters/second
z-component of delta-v     1311.96071903865       meters/second

delta-v magnitude          2707.91086642097       meters/second

heliocentric coordinates of the Earth at departure
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           June  5, 2003

UTC time                14:46:19.786

UTC Julian date         2452796.11550678

TDB time                14:47:23.918

TDB Julian date         2452796.11624905

     sma (au)              eccentricity       inclination (deg)       argper (deg)
 0.100035823926D+01    0.162374015336D-01    0.234390546133D+02    0.102452224545D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.723788828213D-03    0.152048024408D+03    0.254500248953D+03    0.365453189944D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.405615530578D+08    -.134199767646D+09    -.581818397726D+08    0.151789142240D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.282279812575D+02    -.739767150582D+01    -.320740144564D+01    0.293569735091D+02

spacecraft heliocentric coordinates after the first impulse
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           June  5, 2003
```

UTC time 14:46:19.786

UTC Julian date 2452796.11550678

TDB time 14:47:23.918

TDB Julian date 2452796.11624905

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.125929041710D+01	0.194277242259D+00	0.234900279793D+02	0.253491140553D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.455891057237D+00	0.591580038640D+00	0.254082720591D+03	0.516163481019D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.405615530578D+08	-.134199767646D+09	-.581818397726D+08	0.151789142240D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.311238931702D+02	-.792807327794D+01	-.355310213229D+01	0.323137061746D+02

spacecraft heliocentric coordinates prior to the second impulse
(Earth mean equator and equinox of J2000)
---

calendar date December 24, 2003

UTC time 15:23:16.463

UTC Julian date 2452998.14116276

TDB time 15:24:20.595

TDB Julian date 2452998.14190503

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.125929041710D+01	0.194277242259D+00	0.234900279793D+02	0.253491140553D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.455891057237D+00	0.152910259883D+03	0.464014004360D+02	0.516163481019D+03

rx (km)	ry (km)	rz (km)	rmag (km)
0.149989634184D+09	0.146777512084D+09	0.632696170860D+08	0.219188441443D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.146794823540D+02	0.156262551792D+02	0.684180266331D+01	0.225050509173D+02

spacecraft heliocentric coordinates after the second impulse
(Earth mean equator and equinox of J2000)
---

calendar date December 24, 2003

UTC time 15:23:16.463

UTC Julian date 2452998.14116276

TDB time 15:24:20.595

TDB Julian date 2452998.14190503

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.152368015629D+01	0.935418896793D-01	0.246772249523D+02	0.332979237037D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.337165832669D+01	0.707599727739D+02	0.437392098113D+02	0.686972170792D+03

```
     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.149989634184D+09    0.146777512084D+09    0.632696170860D+08    0.219188441443D+09

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 -.167424936383D+02    0.167905252393D+02    0.815376338235D+01    0.250742235839D+02

heliocentric coordinates of Mars at arrival
(Earth mean equator and equinox of J2000)
-----------------------------------------

calendar date           June  5, 2003

UTC time                14:46:19.786

UTC Julian date         2452796.11550678

TDB time                14:47:23.918

TDB Julian date         2452796.11624905

     sma (au)              eccentricity       inclination (deg)       argper (deg)
 0.152368015629D+01    0.935418896802D-01    0.246772249523D+02    0.332979237037D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.337165832666D+01    0.707599727737D+02    0.437392098108D+02    0.686972170792D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.149989634185D+09    0.146777512083D+09    0.632696170854D+08    0.219188441443D+09

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 -.167424936382D+02    0.167905252395D+02    0.815376338242D+01    0.250742235840D+02
```

The following numerical output summarizes the orbital characteristics of the initial circular park orbit and the departure hyperbola.

```
-------------------------------------------------
park orbit and departure hyperbola characteristics
(Earth mean equator and equinox of J2000)
-------------------------------------------------

park orbit
----------

calendar date           June  5, 2003

UTC time                14:46:19.786

UTC Julian date         2452796.11550678

TDB time                14:47:23.918

TDB Julian date         2452796.11624905

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.656346000000D+04    0.746234829302D-16    0.285000000000D+02    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.157131273874D+03    0.432673178863D+02    0.432673178863D+02    0.146996753813D+01

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.594001024866D+04    -.178538269909D+04    0.214655399076D+04    0.656346000000D+04

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 0.298346723698D+01    -.667066112461D+01    0.270766215332D+01    0.779296034444D+01
```

```
departure hyperbola
-------------------

c3                   8.78714081093365       km**2/sec**2

v-infinity           2964.31118658849       meters/second

decl-asymptote      -6.69712585591636       degrees

rasc-asymptote       349.621008346580       degrees

calendar date           June  5, 2003

UTC time                14:46:19.786

UTC Julian date         2452796.11550678

TDB time                14:47:23.918

TDB Julian date         2452796.11624905

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.453617906070D+05    0.114469137819D+01    0.285000000000D+02    0.432673178863D+02

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.157131273874D+03    0.360000000000D+03    0.432673178863D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.594001024866D+04    -.178538269909D+04    0.214655399076D+04    0.656346000000D+04
     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.436921761545D+01    -.976902636339D+01    0.396530755570D+01    0.114126071811D+02

hyperbolic injection delta-v vector and magnitude
(Earth mean equator and equinox of J2000)
-----------------------------------------

delta-vx             1385.75037847178       meters/seconds
delta-vx            -3098.36523877646       meters/seconds
delta-vx             1257.64540237979       meters/seconds

delta-v magnitude    3619.64683669830       meters/seconds
```

The following program output summarizes the flight conditions determined by the n-body, numerically integrated optimal solution.

```
=======================
optimal n-body solution
=======================

orbital element targeting

-------------------------------------------------
park orbit and departure hyperbola characteristics
(Earth mean equator and equinox of J2000)
-------------------------------------------------

park orbit
----------

calendar date           June  5, 2003

UTC time                14:46:19.786

UTC Julian date         2452796.11550678

TDB time                14:47:23.918
```

```
TDB Julian date         2452796.11624905

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.656346000000D+04    0.271163421920D-15    0.285000000000D+02    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.157125111889D+03    0.436020699909D+02    0.436020699909D+02    0.146996753813D+01

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.592542356778D+04    -.181753942725D+04    0.215984099979D+04    0.656346000000D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.302390571929D+01    -.665848743356D+01    0.269272542752D+01    0.779296034444D+01

departure hyperbola
-------------------

c3                   8.78410346610767       km**2/sec**2

v-infinity           2963.79882348780       meters/second

decl-asymptote      -6.85507640009954       degrees

rasc-asymptote       349.917191310227       degrees

calendar date           June  5, 2003

UTC time                14:46:19.786

UTC Julian date         2452796.11550678

TDB time                14:47:23.918

TDB Julian date         2452796.11624905

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.453774756909D+05    0.114464136447D+01    0.285000000000D+02    0.436020699909D+02

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.157125111889D+03    0.254444374517D-13    0.436020699909D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.592542356778D+04    -.181753942725D+04    0.215984099979D+04    0.656346000000D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.442838718653D+01    -.975108458055D+01    0.394338709174D+01    0.114124741106D+02

hyperbolic injection delta-v vector and magnitude
(Earth mean equator and equinox of J2000)
-----------------------------------------

delta-vx             1404.48146723744       meters/seconds
delta-vx            -3092.59714699502       meters/seconds
delta-vx             1250.66166421328       meters/seconds

delta-v magnitude    3619.51376620505       meters/seconds

transfer time        200.582280824892       days


time and conditions at Mars closest approach
(Mars mean equator and IAU node of epoch)
-----------------------------------------

calendar date           December 23, 2003

UTC time                04:44:48.849
```

```
UTC Julian date         2452996.69778760

TDB time                04:45:52.981

TDB Julian date         2452996.69852987

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.582352684004D+04    0.185858625093D+01    0.599999999549D+02    0.113684894654D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.106057738224D+03    0.751920423154D-05    0.113684902174D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.164452397524D+04    -.256343382717D+04    0.396539388663D+04    0.500000007675D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.220854511594D+01    -.407978431915D+01    -.172145562882D+01    0.494830487156D+01
```

B-plane coordinates at Mars closest approach
(Mars mean equator and IAU node of epoch)

```
b-magnitude          9123.33656409976       kilometers
b dot r             -7877.12099850008
b dot t              4602.85072936640
theta                300.299093776394       degrees
v-infinity           2.71189433424749       km/sec
r-periapsis          5000.00007675185       kilometers
decl-asymptote       7.67016256576066       degrees
rasc-asymptote       281.598221796986       degrees

flight path angle   4.888811583587831E-006  degrees

heliocentric coordinates of Mars at closest approach
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           December 23, 2003
UTC time                04:44:48.849

UTC Julian date         2452996.69778760

TDB time                04:45:52.981

TDB Julian date         2452996.69852987

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.152368088274D+01    0.935423938284D-01    0.246772247953D+02    0.332979383120D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.337165797218D+01    0.699444364551D+02    0.429238195751D+02    0.686972662090D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.152062765860D+09    0.144669272600D+09    0.622466060276D+08    0.218922185165D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.165046942915D+02    0.170199793761D+02    0.825258149788D+01    0.251035800833D+02

spacecraft heliocentric coordinates at closest approach
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           December 23, 2003

UTC time                04:44:48.849

UTC Julian date         2452996.69778760
```

```
TDB time                04:45:52.981

TDB Julian date         2452996.69852987

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.123325319525D+01    0.237019944260D+00    0.189958205359D+02    0.271188748496D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.344081269086D+03    0.149685343745D+03    0.608740922404D+02    0.500238140038D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.152064939116D+09    0.144665069265D+09    0.622482212455D+08    0.218921376348D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.133803392484D+02    0.171618530394D+02    0.441800218195D+01    0.222054367538D+02
```

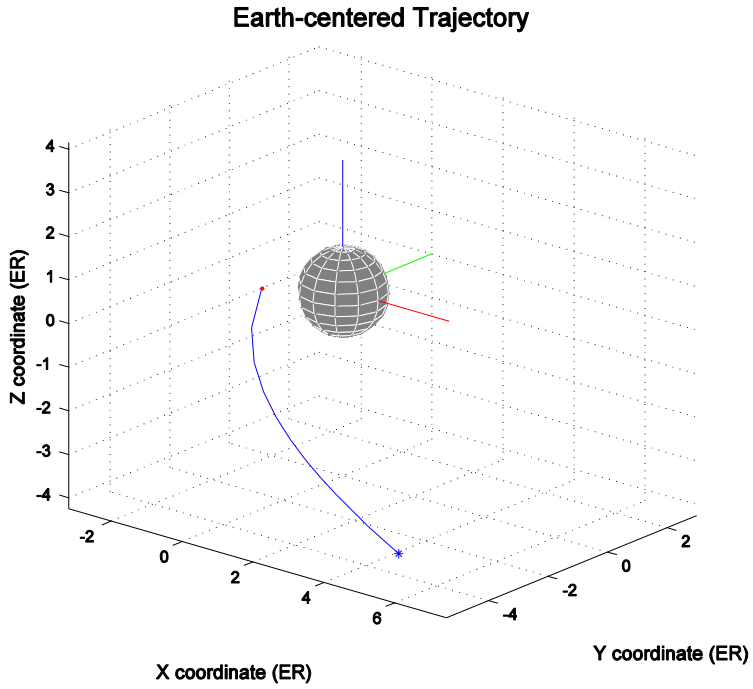
The final program output summarizes both the geocentric and heliocentric spacecraft trajectory characteristics at the Earth's sphere-of-influence (SOI).

```
spacecraft geocentric coordinates at Earth SOI
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           June  8, 2003

UTC time                18:18:25.295

UTC Julian date         2452799.26279277

TDB time                18:19:29.427

TDB Julian date         2452799.26353504

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.455342476095D+05    0.114482949663D+01    0.287299136766D+02    0.437005781805D+02

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.157129617461D+03    0.149331814404D+03    0.193032392585D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.901417895543D+06    -.181707771199D+06    -.100264966368D+06    0.925000000000D+06

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.303091418106D+01    -.537927535773D+00    -.374021744163D+00    0.310091906889D+01

spacecraft heliocentric coordinates at Earth SOI
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           June  8, 2003

UTC time                18:18:25.295

UTC Julian date         2452799.26279277

TDB time                18:19:29.427

TDB Julian date         2452799.26353504

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.127488331145D+01    0.204077670010D+00    0.234949186064D+02    0.253480998747D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.552657115963D+00    0.387349121323D+01    0.257354489961D+03    0.525780022991D+03
```

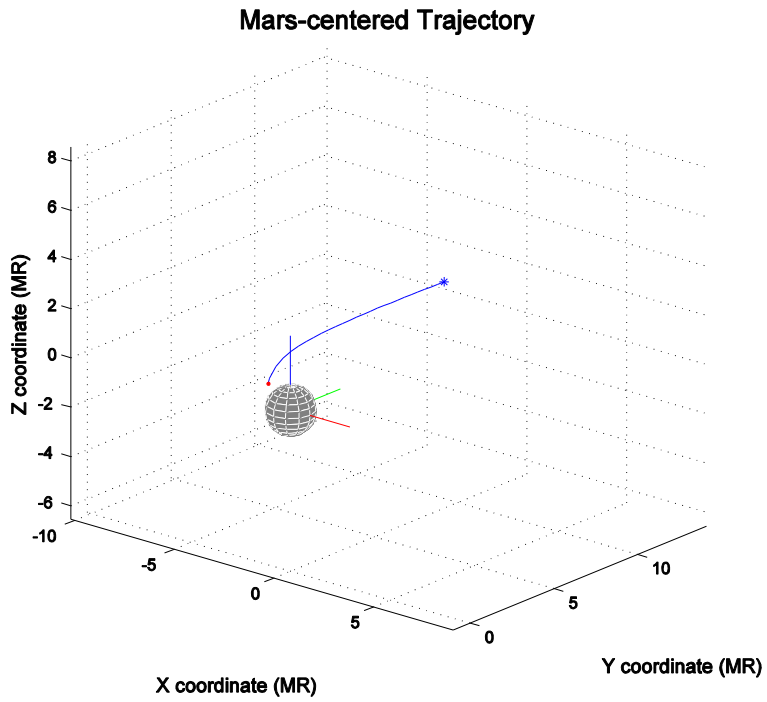
```
     rx (km)                 ry (km)                 rz (km)                rmag (km)
-.319320050194D+08      -.136203485017D+09      -.590719534469D+08      0.151856965462D+09

     vx (kps)                vy (kps)                vz (kps)               vmag (kps)
0.316290976396D+02      -.653910559413D+01      -.297508247213D+01      0.324347165115D+02
```

The `e2m_ftn` computer program will also create three comma-separated-variable (csv) data files. These files summarize the geocentric (`e2m_geo.csv`), heliocentric (`e2m_helio.csv`) and areocentric (`e2m_areo.csv`) trajectory characteristics of the spacecraft.

The following is a graphics display of the geocentric departure trajectory for this example for motion within 30,000 kilometers of the Earth. Injection occurs at perigee of the departure hyperbola which is denoted by the small red dot. The EME2000 x-axis (red), y-axis (green) and z-axis (blue) are also shown on the Earth. The coordinates are displayed in Earth radii (ER).



The following is a screen display of the areocentric or Mars-centered trajectory within 50,000 kilometers of Mars. Closest approach occurs at perigee of the approach hyperbola which is labeled with a small red dot. The x-axis (red), y-axis (green) and z-axis (blue) are also shown relative to the mean equator of Mars and the IAU node of epoch. The coordinates are displayed in Mars radii (MR).



Technical discussion

Designing the departure hyperbola

This section describes the algorithm used to determine the *Earth-centered-inertial* (ECI) state vector of a departure hyperbola for interplanetary missions. In the discussion that follows, interplanetary injection is assumed to occur *impulsively* at perigee of the departure hyperbola.

The departure trajectory for interplanetary missions can be defined using the specific (per unit mass) orbital energy C_3, and the right ascension α_∞ (RLA) and declination δ_∞ (DLA) of the outgoing asymptote. The perigee radius of the departure hyperbola is calculated from the user's value for the altitude of the circular park orbit and the radius of the Earth.

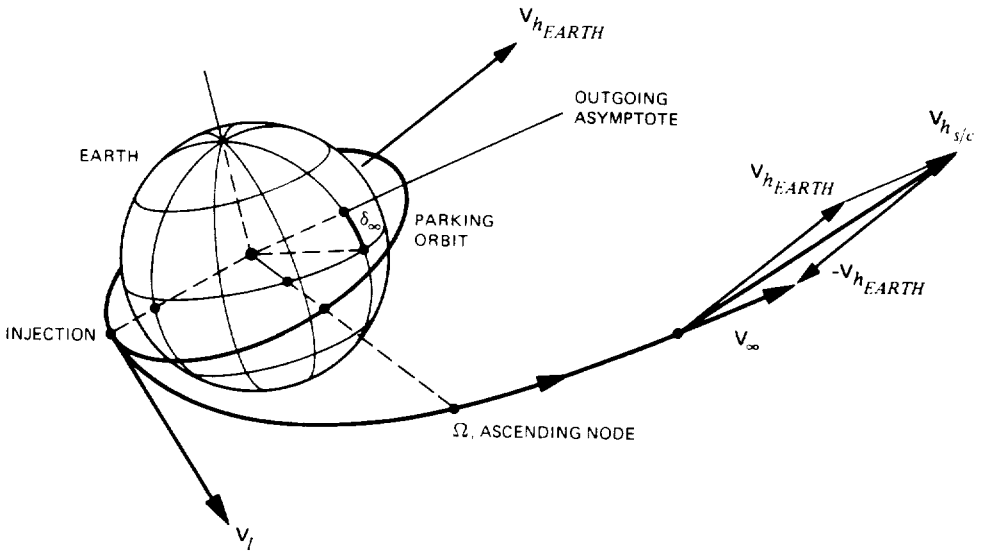
The following figure illustrates the geometry of interplanetary injection. In this diagram $\mathbf{V}_{h_{EARTH}}$ is the heliocentric velocity vector of the Earth at departure, $\mathbf{V}_{h_{s/c}}$ is the heliocentric velocity vector of the spacecraft at departure, $\mathbf{V}_\infty$ is the geocentric v-infinity or *excess velocity* vector and $\mathbf{V}_l$ is the velocity vector of the departure hyperbola at perigee.

From the velocity vector diagram, $\mathbf{V}_\infty = \mathbf{V}_{h_{s/c}} - \mathbf{V}_{h_{EARTH}}$. The heliocentric velocity vector of the spacecraft at departure is determined from the two-body Lambert solution for the heliocentric transfer trajectory. The heliocentric velocity vector of the Earth at departure is determined from a JPL ephemeris.

The two-body Lambert solution is used to initialize the n-body integrated solution by assuming the orientation of the outgoing asymptote of the departure hyperbola can be determined from the components of the geocentric $\mathbf{V}_\infty$ velocity vector.

The twice specific (per unit mass) orbital energy of the departure hyperbola is $C_3 = V_\infty^2$. The right ascension α_∞ and declination δ_∞ of the departure asymptote can be determined from the components of the v-infinity vector according to $\alpha_\infty = \tan^{-1}\left(V_{\infty_y}, V_{\infty_x}\right)$ and $\delta_\infty = \sin^{-1}\left(V_{\infty_z}/V_\infty\right)$. The inverse tangent used in the first equation is a four-quadrant calculation.

Since the right ascension of the outgoing asymptote is an *inertial* coordinate, it should <u>not</u> be called the longitude of the asymptote. This terminology also applies to the right ascension of the ascending node.



This figure was extracted from page 2 of *Interplanetary Mission Design Handbook, Volume 1, Part 2*, JPL Publication 82-43, September 15, 1983.

The orbital inclination is computed from the user-defined launch azimuth $\sum_L$ (measured positive clockwise from north) and launch site geocentric latitude ϕ_L using this equation

$$i = \cos^{-1}\left(\cos\phi_L \sin\sum_L\right)$$

The algorithm used to design the departure hyperbola is valid for geocentric orbit inclinations that satisfy the following inequality constraint

$$|i| > |\delta_\infty|$$

If this inequality is not satisfied, the software will print the following error message

```
park orbit error!!

|inclination| must be > |asymptote declination|
```

The code will also print the inclination of the park orbit, the declination of the departure hyperbola and pause. The user can then change either the azimuth, launch site latitude or orbital inclination to satisfy this constraint and restart the script.

A unit vector in the direction of the departure asymptote is given by

$$\hat{\mathbf{S}} = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

where

$\alpha_\infty =$ right ascension of departure asymptote (RLA)

$\delta_\infty =$ declination of departure asymptote (DLA)

The T-axis direction of the B-plane coordinate system is determined from the following vector cross product

$$\hat{\mathbf{T}} = \hat{\mathbf{S}} \times \hat{\mathbf{u}}_z$$

where $\hat{\mathbf{u}}_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ is a unit vector perpendicular to the Earth's equatorial plane.

The following cross product operation completes the B-plane coordinate system.

$$\hat{\mathbf{R}} = \hat{\mathbf{S}} \times \hat{\mathbf{T}}$$

The B-plane angle is determined from the orbital inclination of the departure hyperbola *i* and the declination of the outgoing asymptote according to

$$\cos\theta = \cos i / \cos\delta_\infty$$

The unit angular momentum vector of the departure hyperbola is given by

$$\hat{\mathbf{h}} = \hat{\mathbf{T}} \sin\theta - \hat{\mathbf{R}} \cos\theta$$

The sine and cosine of the true anomaly at infinity are given by the next two equations

$$\cos\theta_\infty = -\frac{\mu}{r_p V_\infty^2 + \mu} \qquad \sin\theta_\infty = \sqrt{1 - \cos^2\theta_\infty}$$

where $V_\infty = \sqrt{C_3} = V_L - V_p$ is the spacecraft's velocity at infinity (v-infinity), V_L is the heliocentric departure velocity determined from the Lambert solution, V_p is the heliocentric velocity of the departure planet, and r_p is the user-specified perigee radius of the departure hyperbola.

A unit vector in the direction of perigee of the departure hyperbola is determined from

$$\hat{\mathbf{r}}_p = \hat{\mathbf{S}} \cos\theta_\infty - \left(\hat{\mathbf{h}} \times \hat{\mathbf{S}}\right) \sin\theta_\infty$$

The ECI position vector at perigee is equal to $\mathbf{r}_p = r_p \hat{\mathbf{r}}_p$.

The scalar magnitude of the departure hyperbola perigee velocity can be determined from

$$V_p = \sqrt{\frac{2\mu}{r_p} + V_\infty^2} = \sqrt{V_{lc}^2 + V_\infty^2}$$

where $V_{lc} = \sqrt{\mu / r_p}$ is the local circular velocity.

A unit vector aligned with the velocity vector at perigee is $\hat{\mathbf{v}}_p = \hat{\mathbf{h}} \times \hat{\mathbf{r}}_p$.

The ECI velocity vector at perigee of the departure hyperbola is given by

$$\mathbf{v}_p = V_p \hat{\mathbf{v}}_p$$

Finally, the classical orbital elements of the departure hyperbola can be determined from the position and velocity vectors at perigee. The impulsive injection delta-v vector and magnitude can be determined from the velocity difference between the local circular velocity of the park orbit and the geocentric velocity of the departure hyperbola each evaluated at the orbital location of the propulsive maneuver.

Capture maneuver

Typically, a retrograde propulsive maneuver is performed at periapsis to create an elliptical orbit about Mars. At the next or any subsequent apoapsis, an efficient plane change maneuver can be performed to establish the orbital inclination of the mission orbit. Finally, one or more propulsive maneuvers at periapsis will establish a circular or elliptical mission orbit.

The velocity decrement required to establish an elliptical *capture* orbit of eccentricity *e* is given by

$$\Delta v = \sqrt{v_\infty^2 + \frac{2\mu}{r_p}} - \sqrt{\frac{\mu(1+e)}{r_p}} = \sqrt{v_\infty^2 + \frac{2\mu}{r_p}} - \sqrt{\frac{2\mu r_a}{r_p(r_a + r_p)}}$$

where r_p and r_a are the periapsis and apoapsis radii of the capture orbit.

For a <u>circular</u> capture orbit of radius *r*, the impulsive delta-v equation simplifies to

$$\Delta v = \sqrt{v_\infty^2 + \frac{2\mu}{r}} - \sqrt{\frac{\mu}{r}} = \sqrt{v_\infty^2 + \frac{2\mu}{r}} - v_{lc}$$

where v_{lc} is the "local circular" speed of the capture orbit.

If we set the partial derivative $\partial \Delta v / \partial r_a = 0$ from the elliptical orbit capture expression, we find a "best" capture apoapsis radius and corresponding minimum delta-v according to

$$r_a = \frac{2\mu}{v_\infty^2} \qquad \Delta v = v_\infty \sqrt{\frac{1-e}{2}}$$

where $e = r_a - r_p / r_a + r_p$ is the orbital eccentricity of the elliptical capture orbit. Notice the optimum apoapsis radius is dependent only on the incoming v-infinity and the planet's gravity.

Likewise, the optimum capture periapsis radius and minimum capture delta-v is

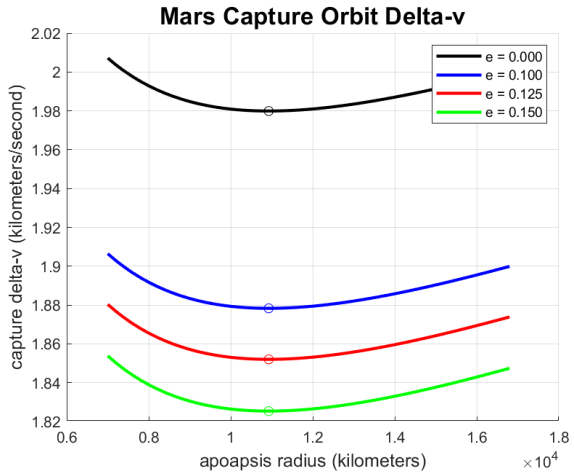
$$r_p = \frac{2\mu}{v_\infty^2}\left[\frac{1-e}{1+e}\right] \qquad \Delta v = \sqrt{v_\infty^2 + \frac{2\mu}{r_p}} - \sqrt{\frac{\mu(1+e)}{r_p}}$$

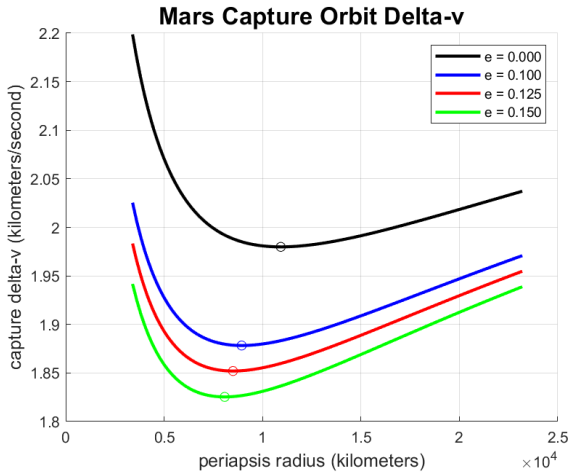
For a circular capture orbit, the optimum periapsis and apoapsis radii are identical and the minimum delta-v is as follows

$$r_a = r_p = \frac{2\mu}{v_\infty^2} \qquad \Delta v = \sqrt{v_\infty^2 + \frac{2\mu}{r}} - \sqrt{\frac{\mu}{r}}$$

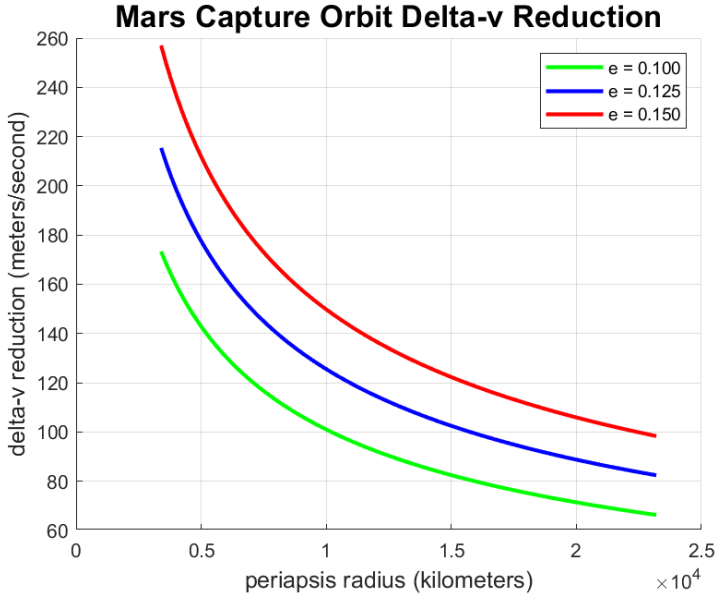
The following two plots illustrate the behavior of capture delta-v as a function of the eccentricity of the capture orbit. Each line of the graphs is labeled with a small circle to indicate the optimum radius and impulsive capture delta-v. The v-infinity for this example is 2.8 kilometers meters/second.

In the graph on the left, we can see that the optimum apoapsis radius is independent of orbital eccentricity.





The reduction in delta-v required for capture between a circular and elliptical orbit is given by $\Delta(\Delta v) = v_{lc}\left(\sqrt{1+e}-1\right)$ where v_{lc} is the local circular velocity evaluated at the periapsis of the capture orbit and *e* is the orbital eccentricity of the capture orbit. The following is a graph of the impulsive delta-v reduction in meters/second as a function of the capture orbit periapsis radius and orbital eccentricity.



As an exercise, see if you can derive the expression for the optimum circular orbit radius r_c from

$$\frac{\partial}{\partial r_c}\Delta v = -\left(v_\infty^2 + \frac{2\mu}{r_c}\right)^{-1/2} \mu r_c^{-2} + \frac{1}{2}\sqrt{\mu} r_c^{-3/2} = 0$$

Hint: multiply the expression by $r_c^{3/2}$, rearrange and square the result. Solve for r_c.

Questions: are these equations and results valid as well for the Earth departure hyperbolic trajectory? Is there a reasonable optimal circular orbit altitude that minimizes the interplanetary injection delta-v?

Propagating the spacecraft's trajectory

The spacecraft's orbital motion is modeled with respect to the Earth mean equator and equinox of J2000 (EME2000) coordinate system described in *Appendix C – Cartesian Equations of Motion*.

Heliocentric trajectory propagation

The second-order, vector system of heliocentric equations of motion for *point-mass* gravity perturbations such as the Moon or planets are given by

$$\ddot{\mathbf{r}} = -\sum_{j=1}^{n} \mu_j \left[\frac{\mathbf{d}_j}{d_j^3} + \frac{\mathbf{s}_j}{s_j^3}\right]$$

In this equation, $\mathbf{s}_j$ is the vector from the primary body to the secondary body *j*, μ_j is the gravitational constant of the secondary body, and $\mathbf{d}_j = \mathbf{r} - \mathbf{s}_j$, where $\mathbf{r}$ is the position vector of the spacecraft relative to the primary body.

Following the geocentric formulation, use is again made of Professor Battin's $F(q)$ function given by

$$F(q_k) = q_k \left[\frac{3 + 3q_k + q_k^2}{1 + \left(\sqrt{1+q_k}\right)^3}\right]$$

where

$$q_k = \frac{\mathbf{r}^T(\mathbf{r} - 2\mathbf{s}_k)}{\mathbf{s}_k^T \mathbf{s}_k}$$

The third-body acceleration can now be expressed as

$$\ddot{\mathbf{r}} = -\sum_{k=1}^{n} \frac{\mu_k}{d_k^3}\left[\mathbf{r} + F(q_k)\mathbf{s}_k\right]$$

Heliocentric acceleration due to solar radiation pressure

The heliocentric acceleration vector of the spacecraft due to solar radiation pressure is given by:

$$\mathbf{a}_{srp} = c_{srp} \frac{\mathbf{r}_{sc}}{|\mathbf{r}_{sc}|^3}$$

where $\mathbf{r}_{sc}$ is the heliocentric position vector of the spacecraft. The equation for c_{srp} is defined in the previous geocentric trajectory propagation discussion.

The B-plane

A description of the B-plane geometry and important equations can be found in *Appendix D – B-Plane Geometry, Coordinates and Targeting*.

Predicting the flight conditions at the Earth's sphere of influence

The trajectory conditions at the boundary of the Earth's sphere of influence are determined during the numerical integration of the spacecraft's geocentric equations of motion by finding the time at which the difference between the geocentric distance and the user-defined value is essentially zero. This scalar mission constraint is $\Delta r = \left|\mathbf{r}_{sc}\right|_p - r_{soi_u} \approx 0$ where $\left|\mathbf{r}_{sc}\right|_p$ is the scalar magnitude of the *predicted* geocentric position vector of the spacecraft and r_{soi_u} is the user-defined value of the geocentric distance of the SOI boundary.

Targeting to the Mars-centered flight path angle, periapsis radius and orbital inclination

The software solves the B-plane targeting problem by minimizing the magnitude of the hyperbolic injection maneuver while satisfying two nonlinear *equality constraint* equations. These constraint equations are the differences between components of the *required* B-plane and the B-plane components *predicted* by the software.

The predicted flight path angle is $\gamma_p = \sin^{-1}\left(\mathbf{r}\bullet\mathbf{v}/\left|\mathbf{r}\bullet\mathbf{v}\right|\right)$ where **r** and **v** are the Mars-centered position and velocity vectors, respectively.

Both the geocentric SOI boundary and areocentric flight path angle are predicted using a Runge-Kutta-Fehlberg (RKF7(8)) integrator embedded within a one-dimensional derivative-free form of Brent's root-finding method.

For this targeting option, the following series of equations can be used to determine the required B-plane target vector

$$\mathbf{B}\bullet\mathbf{T} = b_t \cos\theta \qquad \mathbf{B}\bullet\mathbf{R} = b_t \sin\theta$$

where

$$b_t = \cos\gamma_u \sqrt{\frac{2\mu r_u}{v_\infty^2} + r_u^2}$$

In these equations, γ_u is the user-defined flight path angle, and r_u is the user-defined Mars-centered radius at the entry interface. Note that a user-defined flight path angle equal to zero will predict closest approach at Mars. Furthermore, for this case *r* will be equal to the periapsis radius of the incoming areocentric hyperbola.

Also, from the user-defined areocentric (Mars-centered) inclination i_u,

$$\cos\theta = \frac{\cos i_u}{\cos\delta_\infty} \qquad \sin\theta = -\sqrt{1-\cos^2\theta}$$

The areocentric declination of the incoming hyperbola can be determined from

$$\sin\delta_\infty = \left|\hat{\mathbf{s}} \times \hat{\mathbf{z}}\right| = \sqrt{s_x^2 + s_y^2}$$

$$\hat{\mathbf{z}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

The arrival asymptote unit vector $\hat{\mathbf{S}}$ is given by

$$\hat{\mathbf{S}} = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix} = \begin{Bmatrix} s_x \\ s_y \\ s_z \end{Bmatrix}$$

where δ_∞ and α_∞ are the declination and right ascension of the asymptote of the incoming hyperbola at Mars.

The nonlinear equality mission constraints enforced by the nonlinear programming algorithm are

$$\left(\mathbf{B}\bullet\mathbf{T}\right)_p - \left(\mathbf{B}\bullet\mathbf{T}\right)_r \approx 0 \qquad \left(\mathbf{B}\bullet\mathbf{R}\right)_p - \left(\mathbf{B}\bullet\mathbf{R}\right)_r \approx 0$$

where the *p* subscript refers to coordinates *predicted* by the software and the *r* subscript denotes the *required* B-plane coordinates defined previously.

Important note!!

This technique is valid for aerocentric orbit inclinations that satisfy the following inequality

$$\left|i\right| > \left|\delta_\infty\right|$$

If this requirement is not satisfied, the software will print the following error message

```
b-plane targeting error!!

|inclination| must be > |asymptote declination|
```

It will also display the actual declination of the asymptote and stop. The user should then edit the input file, include a valid orbital inclination, and restart the simulation.

Targeting to user-defined B-plane coordinates

For this targeting option, the nonlinear equality constraints enforced by the nonlinear programming algorithm are

$$\left(\mathbf{B}\bullet\mathbf{T}\right)_p - \left(\mathbf{B}\bullet\mathbf{T}\right)_u = 0 \qquad \left(\mathbf{B}\bullet\mathbf{R}\right)_p - \left(\mathbf{B}\bullet\mathbf{R}\right)_u = 0$$

where the *p* subscript refers to coordinates predicted by the software and the *u* subscript denotes coordinates provided by the user. The *predicted* B-plane coordinates are based on the Mars-centered flight conditions at closest approach.

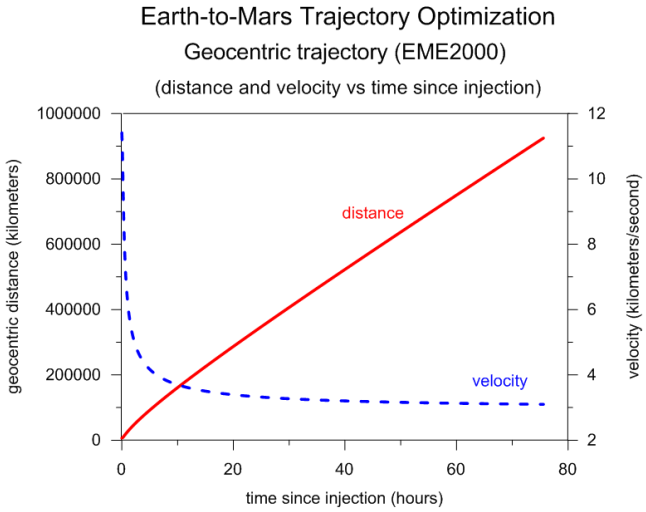
Geocentric-to-areocentric coordinate transformation

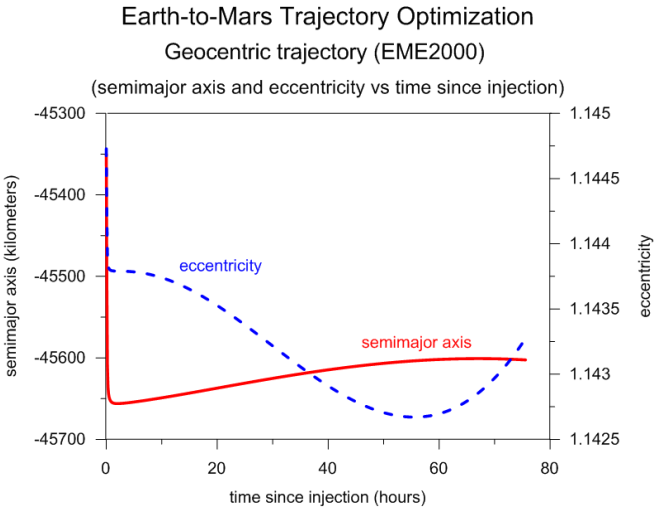
Please consult *Appendix G – Aerospace Trajectory Coordinates and Time Systems* which describes the relationship between geocentric and areocentric coordinate systems.

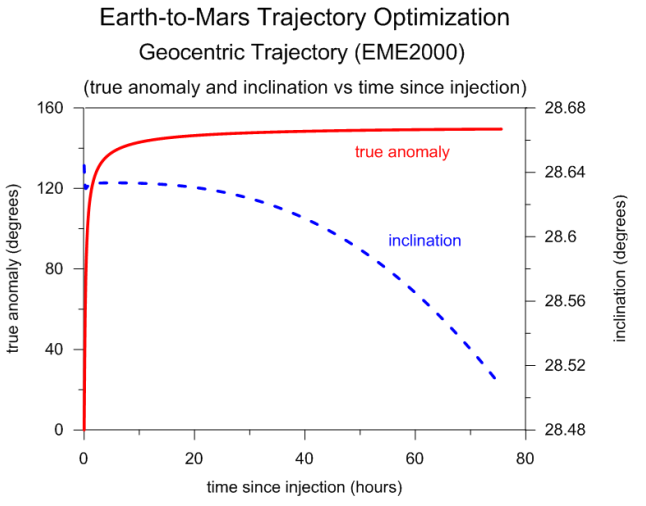
Graphic displays of flight characteristics

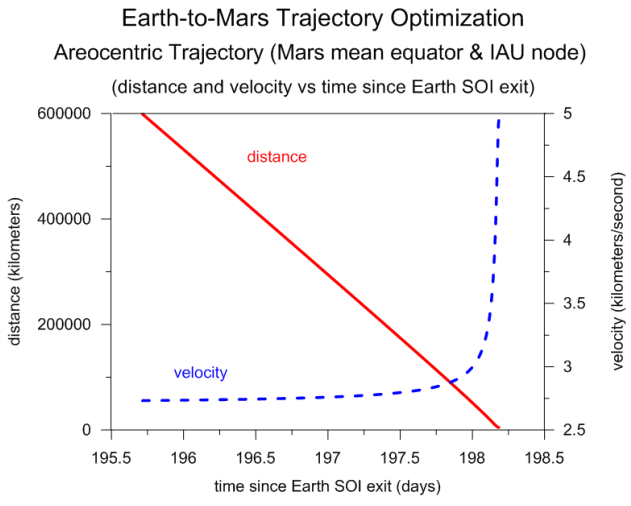
This section contains typical graphic displays of important flight characteristics during the different phases of the mission. The geocentric and heliocentric plots are relative to the EME2000 coordinate system and the areocentric graphs are relative to the Mars mean equator and IAU node of epoch.

The first three plots are geocentric flight parameters as a function of time since the hyperbolic injection. The data displays terminate at exit from the Earth's sphere-of-influence (SOI).

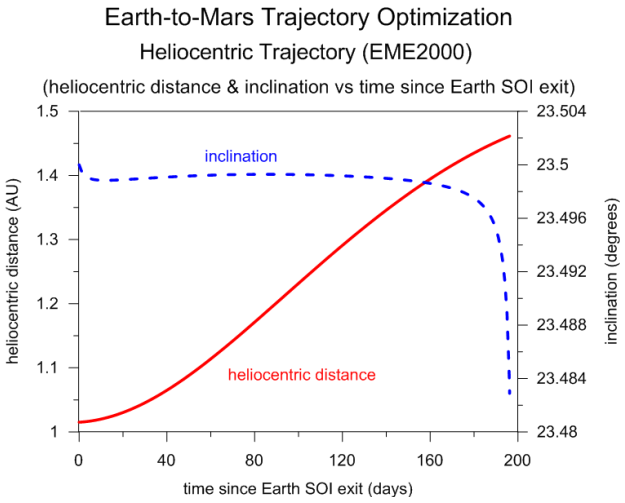


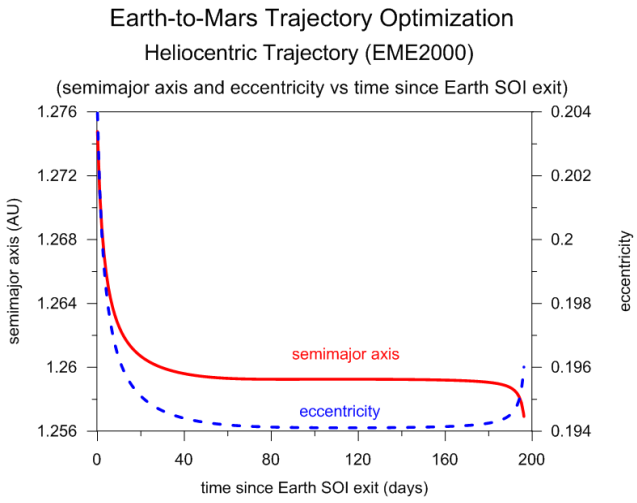




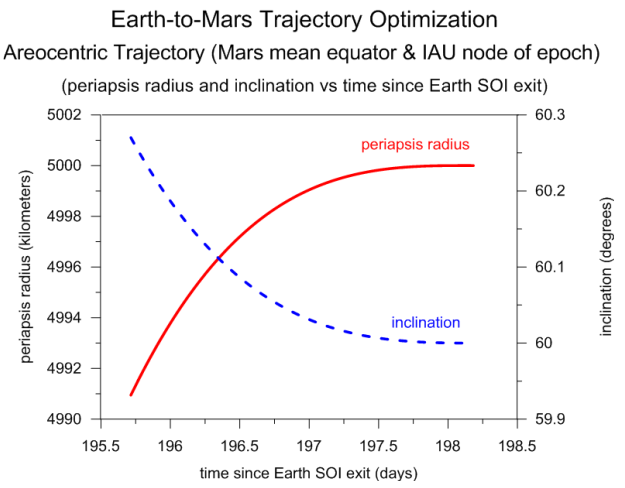


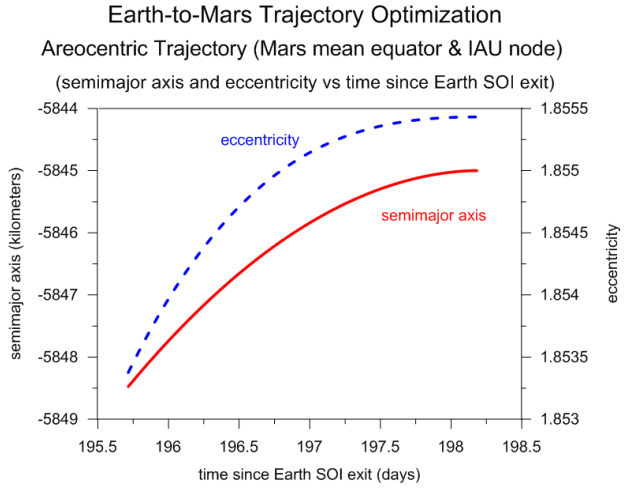
These next two plots are heliocentric trajectory characteristics as a function of time since exit from the Earth's sphere-of-influence (SOI). The data terminates when the spacecraft is within two days of closest approach to Mars.





The final two plots are Mars-centered trajectory characteristics as a function of time since exit from the Earth's SOI. The data display starts when the spacecraft is 600,000 kilometers from Mars and ends at closest approach to Mars. The orbital inclination is relative to the mean equator of Mars.





Mars entry interface example

The following is the `e2m_ftn` optimal n-body program output for a typical entry interface example. For this example, the areocentric flight path angle is -2 degrees, the areocentric radius is 3500 kilometers, and the areocentric orbital inclination target is 45 degrees.

```
=======================
optimal n-body solution
=======================

orbital element targeting

--------------------------------------------------
park orbit and departure hyperbola characteristics
(Earth mean equator and equinox of J2000)
--------------------------------------------------
```

park orbit
---

calendar date June 5, 2003

UTC time 14:46:19.786

UTC Julian date 2452796.11550678

TDB time 14:47:23.918

TDB Julian date 2452796.11624905

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.656346000000D+04	0.110917091772D-15	0.286442848562D+02	0.000000000000D+00

raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.270931886661D+01	0.194727110636D+03	0.194727110636D+03	0.146996753813D+01

rx (km)	ry (km)	rz (km)	rmag (km)
-.627152167852D+04	-.176274500778D+04	-.799845638258D+03	0.656346000000D+04

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.229153863954D+01	-.651347903009D+01	-.361298664793D+01	0.779296034444D+01

departure hyperbola
---

c3	8.79578987310781	km**2/sec**2
v-infinity	2965.76969320071	meters/second
decl-asymptote	-6.84967869961631	degrees
rasc-asymptote	350.005266398407	degrees

calendar date June 5, 2003

UTC time 14:46:19.786

UTC Julian date 2452796.11550678

TDB time 14:47:23.918

TDB Julian date 2452796.11624905

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.453171855229D+05	0.114483379593D+01	0.286442848562D+02	0.194727110636D+03

raan (deg)	true anomaly (deg)	arglat (deg)
0.270931886661D+01	0.000000000000D+00	0.194727110636D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.627152167852D+04	-.176274500778D+04	-.799845638258D+03	0.656346000000D+04

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.335601587665D+01	-.953915358882D+01	-.529130966565D+01	0.114129861006D+02

hyperbolic injection delta-v vector and magnitude
(Earth mean equator and equinox of J2000)
---

delta-vx	1064.47723711808	meters/seconds
delta-vx	-3025.67455872508	meters/seconds
delta-vx	-1678.32301772060	meters/seconds
delta-v magnitude	3620.02575618869	meters/seconds
transfer time	201.329916741233	days

time and conditions at Mars entry interface
(Mars mean equator and IAU node of epoch)
---

calendar date December 23, 2003

```
UTC time                22:41:24.592

UTC Julian date         2452997.44542352

TDB time                22:42:28.724

TDB Julian date         2452997.44616579

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.584592533218D+04    0.159811483058D+01    0.450000001093D+02    0.118029275635D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.108956668867D+03    0.356748680746D+03    0.114777956381D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.164865109527D+04    -.211725961303D+04    0.224703384781D+04    0.349999997373D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.333447912072D+01    -.417773239851D+01    -.179648318722D+01    0.563910904683D+01
```

B-plane coordinates at Mars entry interface
(Mars mean equator and IAU node of epoch)

```
b-magnitude          7287.43544300997       kilometers
b dot r             -5107.54726266006
b dot t              5198.04543032587
theta                315.503127761648       degrees
v-infinity           2.70669407634656       km/sec
r-periapsis          3496.53463966152       kilometers
decl-asymptote       7.54885673068842       degrees
rasc-asymptote       281.341426102540       degrees

flight path angle   -1.99999962370103       degrees

heliocentric coordinates of Mars at entry interface
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           December 23, 2003

UTC time                22:41:24.592

UTC Julian date         2452997.44542352

TDB time                22:42:28.724

TDB Julian date         2452997.44616579

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.152368050974D+01    0.935421335240D-01    0.246772248803D+02    0.332979309023D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.337165816278D+01    0.703671118493D+02    0.433464208718D+02    0.686972409831D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.150992631967D+09    0.145764865767D+09    0.627780413362D+08    0.219059931239D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.166283847981D+02    0.169014372089D+02    0.820155202609D+01    0.250883880781D+02

spacecraft heliocentric coordinates at entry interface
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           December 23, 2003

UTC time                22:41:24.592

UTC Julian date         2452997.44542352

TDB time                22:42:28.724
TDB Julian date         2452997.44616579
```

```
     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.124418233994D+01    0.260193891025D+00    0.185282046360D+02    0.281382377649D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.340795030083D+03    0.143020395729D+03    0.644027733781D+02    0.506902549814D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.150993772736D+09    0.145761597092D+09    0.627785555405D+08    0.219058689916D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.127216565242D+02    0.178536794502D+02    0.424803969529D+01    0.223302542727D+02
```

User-defined initial guess example

This section is a summary of the `e2m_ftn` program output for a simulation which exercised the user-defined initial guess option. The used-defined conditions for this case are as follows.

```
*************************
user-defined initial guess
*************************

user-defined departure calendar date (month, day, year)
6, 5, 2003

user-defined departure UTC (hours, minutes, seconds)
14, 46, 20

user-defined C3 (kilometers^2/second^2)
8.787

user-defined EME2000 RLA (degrees; 0 <= RLA <= 360)
349.621

user-defined EME2000 DLA (degrees; -90 <= DLA <= +90)
-6.697
```

The program output for this case is

```
 *************************
 user-defined initial guess
 *************************

 --------------------------------------------------
 park orbit and departure hyperbola characteristics
 (Earth mean equator and equinox of J2000)
 --------------------------------------------------

 park orbit
 ----------

 calendar date           June  5, 2003

 UTC time                14:46:20.000

 UTC Julian date         2452796.11550926

 TDB time                14:47:24.132

 TDB Julian date         2452796.11625153

      sma (km)             eccentricity       inclination (deg)       argper (deg)
  0.656346000000D+04    0.469828416638D-15    0.285000000000D+02    0.000000000000D+00

     raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
  0.157131506217D+03    0.432668393961D+02    0.432668393961D+02    0.146996753813D+01

      rx (km)                ry (km)              rz (km)              rmag (km)
  -.594002399326D+04    -.178535986763D+04    0.214653494594D+04    0.656346000000D+04
      vx (kps)               vy (kps)             vz (kps)             vmag (kps)
  0.298343538855D+01    -.667066672934D+01    0.270768343765D+01    0.779296034444D+01
```

```
departure hyperbola
-------------------

c3                   8.78700000000000       km**2/sec**2

v-infinity           2964.28743545561       meters/second

decl-asymptote      -6.69700000000000       degrees

rasc-asymptote       349.621000000000       degrees


calendar date           June  5, 2003

UTC time                14:46:20.000

UTC Julian date         2452796.11550926

TDB time                14:47:24.132

TDB Julian date         2452796.11625153

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.453625175259D+05    0.114468905956D+01    0.285000000000D+02    0.432668393961D+02

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.157131506217D+03    0.360000000000D+03    0.432668393961D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.594002399326D+04    -.178535986763D+04    0.214653494594D+04    0.656346000000D+04

     vx (kps)               vy (kps)              vz (kps)             vmag (kps)
 0.436916861242D+01    -.976902929071D+01    0.396533658263D+01    0.114126010120D+02

hyperbolic injection delta-v vector and magnitude
(Earth mean equator and equinox of J2000)
-----------------------------------------

delta-vx             1385.73322386573       meters/seconds
delta-vx            -3098.36256137726       meters/seconds
delta-vx             1257.65314498090       meters/seconds

delta-v magnitude    3619.64066760085       meters/seconds

=======================
optimal n-body solution
=======================

orbital element targeting

--------------------------------------------------
park orbit and departure hyperbola characteristics
(Earth mean equator and equinox of J2000)
--------------------------------------------------

park orbit
----------

calendar date           June  5, 2003

UTC time                14:46:20.000

UTC Julian date         2452796.11550926

TDB time                14:47:24.132

TDB Julian date         2452796.11625153

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.656346000000D+04    0.000000000000D+00    0.285000000000D+02    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.157125107020D+03    0.436020736083D+02    0.436020736083D+02    0.146996753813D+01
```

```
     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.592542356145D+04    -.181753927773D+04    0.215984114297D+04    0.656346000000D+04

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 0.302390559760D+01    -.665848755430D+01    0.269272526562D+01    0.779296034444D+01

departure hyperbola
-------------------

c3                   8.78410344274025       km**2/sec**2

v-infinity           2963.79881954566       meters/second

decl-asymptote      -6.85507809941981       degrees

rasc-asymptote       349.917189696710       degrees


calendar date           June  5, 2003

UTC time                14:46:20.000

UTC Julian date         2452796.11550926

TDB time                14:47:24.132

TDB Julian date         2452796.11625153

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.453774758117D+05    0.114464136408D+01    0.285000000000D+02    0.436020736083D+02

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.157125107020D+03    0.360000000000D+03    0.436020736083D+02

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.592542356145D+04    -.181753927773D+04    0.215984114297D+04    0.656346000000D+04

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 0.442838700792D+01    -.975108475650D+01    0.394338685428D+01    0.114124741096D+02

hyperbolic injection delta-v vector and magnitude
(Earth mean equator and equinox of J2000)
-----------------------------------------

delta-vx             1404.48141031891       meters/seconds
delta-vx            -3092.59720219989       meters/seconds
delta-vx             1250.66158866052       meters/seconds

delta-v magnitude    3619.51376518129       meters/seconds

transfer time        200.582275519148       days

time and conditions at Mars closest approach
(Mars mean equator and IAU node of epoch)
-----------------------------------------

calendar date           December 23, 2003

UTC time                04:44:48.605

UTC Julian date         2452996.69778478

TDB time                04:45:52.737

TDB Julian date         2452996.69852705

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.582352864328D+04    0.185858597496D+01    0.599999996757D+02    0.113684893625D+03
    raan (deg)         true anomaly (deg)       arglat (deg)
 0.106057741177D+03    0.359999992120D+03    0.113684885745D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.164452448225D+04    -.256343270506D+04    0.396539432758D+04    0.500000001793D+04
```

```
     vx (kps)               vy (kps)               vz (kps)            vmag (kps)
 0.220854505835D+01    -.407978431421D+01    -.172145511153D+01    0.494830466183D+01

B-plane coordinates at Mars closest approach
(Mars mean equator and IAU node of epoch)
----------------------------------------

b-magnitude          9123.33748256896       kilometers
b dot r             -7877.12173368605
b dot t              4602.85129170372
theta                300.299094496175       degrees
v-infinity           2.71189391438144       km/sec
r-periapsis          5000.00001792543       kilometers
decl-asymptote       7.67016812143089       degrees
rasc-asymptote       281.598221423417       degrees

flight path angle  -5.123380764659380E-006  degrees

heliocentric coordinates of Mars at closest approach
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           December 23, 2003

UTC time                04:44:48.605

UTC Julian date         2452996.69778478

TDB time                04:45:52.737

TDB Julian date         2452996.69852705

     sma (au)              eccentricity       inclination (deg)       argper (deg)
 0.152368088274D+01    0.935423938294D-01    0.246772247953D+02    0.332979383120D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.337165797218D+01    0.699444348571D+02    0.429238179775D+02    0.686972662091D+03

     rx (km)                ry (km)               rz (km)              rmag (km)
 0.152062769888D+09    0.144669268446D+09    0.622466040136D+08    0.218922184645D+09

     vx (kps)               vy (kps)               vz (kps)            vmag (kps)
 -.165046938221D+02    0.170199798227D+02    0.825258169004D+01    0.251035801406D+02

spacecraft heliocentric coordinates at closest approach
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           December 23, 2003

UTC time                04:44:48.605

UTC Julian date         2452996.69778478

TDB time                04:45:52.737

TDB Julian date         2452996.69852705

     sma (au)              eccentricity       inclination (deg)       argper (deg)
 0.123325318087D+01    0.237019957488D+00    0.189958209111D+02    0.271188742859D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.344081272746D+03    0.149685344344D+03    0.608740872028D+02    0.500238131289D+03

     rx (km)                ry (km)               rz (km)              rmag (km)
 0.152064943143D+09    0.144665065111D+09    0.622482192324D+08    0.218921375828D+09

     vx (kps)               vy (kps)               vz (kps)            vmag (kps)
 -.133803385899D+02    0.171618532359D+02    0.441800278960D+01    0.222054366298D+02


spacecraft geocentric coordinates at Earth SOI
(Earth mean equator and equinox of J2000)
----------------------------------------
```

```
calendar date           June  8, 2003

UTC time                18:18:25.510

UTC Julian date         2452799.26279525

TDB time                18:19:29.642

TDB Julian date         2452799.26353752

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.455342476454D+05    0.114482950045D+01    0.287299125623D+02    0.437005825730D+02

    raan (deg)          true anomaly (deg)       arglat (deg)
 0.157129612151D+03    0.149331814038D+03    0.193032396611D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.901417887538D+06    -.181707796072D+06    -.100264993256D+06    0.925000000000D+06

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.303091415458D+01    -.537927617009D+00    -.374021832727D+00    0.310091906778D+01

spacecraft heliocentric coordinates at Earth SOI
(Earth mean equator and equinox of J2000)
----------------------------------------

calendar date           June  8, 2003

UTC time                18:18:25.510

UTC Julian date         2452799.26279525

TDB time                18:19:29.642

TDB Julian date         2452799.26353752

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.127488331207D+01    0.204077670162D+00    0.234949186214D+02    0.253480998595D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (days)
 0.552657327881D+00    0.387349354483D+01    0.257354492140D+03    0.525780023378D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.319319988936D+08    -.136203486329D+09    -.590719540316D+08    0.151856965578D+09

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.316290978725D+02    -.653910456968D+01    -.297508208067D+01    0.324347164962D+02
```

Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `e2m_ftn` software.

The simulation summary screen display contains the following information.

calendar date = UTC calendar date of trajectory event

UTC time = UTC time of trajectory event

UTC Julian Date = Julian Date of trajectory event on UTC time scale

TDB time = TDB time of trajectory event

TDB Julian Date = Julian Date of trajectory event on TDB time scale

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of periapsis in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (days) = orbital period in days

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (kps) = x-component of the spacecraft's velocity vector in kilometers per second

vy (kps) = y-component of the spacecraft's velocity vector in kilometers per second

vz (ksp) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (kps) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

b-magnitude = magnitude of the b-plane vector

b dot r = dot product of the B-plane b-vector and r-vector

b dot t = dot product of the B-plane b-vector and t-vector

theta = orientation of the b-plane vector in degrees

v-infinity = magnitude of outgoing or incoming v-infinity vector in kilometers/second

r-periapsis = periapsis radius of incoming or outgoing hyperbola in kilometers

decl-asymptote = declination of incoming v-infinity vector in degrees

rasc-asymptote = right ascension of incoming v-infinity vector in degrees

flight path angle = flight path angle in degrees

c3 = specific orbital energy in kilometers squared per seconds squared

launch azimuth = launch azimuth in degrees measure positive clockwise from north

launch latitude = geocentric latitude of the launch site in degrees

delta-vx = x-component of injection delta-v in meters per second

delta-vy = y-component of injection delta-v in meters per second

delta-vz = z-component of injection delta-v in meters per second

delta-v magnitude = scalar magnitude of the injection delta-v in meters per second

transfer time = Lambert two-body solution transfer time in days

The `e2m_geo.csv` disk file contains the following information.

time (hrs) = simulation time since interplanetary injection in hours

re2sc-x (km) = x-component of the spacecraft's geocentric position vector in kilometers

re2sc-y (km) = y-component of the spacecraft's geocentric position vector in kilometers

re2sc-z (km) = z-component of the spacecraft's geocentric position vector in kilometers

re2sc-mag (km) = the spacecraft's geocentric radius in kilometers

ve2sc-y (km/sec) = y-component of the spacecraft's geocentric velocity vector in kilometers per second

ve2sc-y (km/sec) = y-component of the spacecraft's geocentric velocity vector in kilometers per second

ve2sc-z (km/sec) = z-component of the spacecraft's geocentric position vector in kilometers per second

ve2sc-mag (km/sec) = the spacecraft's geocentric speed in kilometers per second

sma-geo (km) = geocentric semimajor axis of the spacecraft in kilometers

ecc-geo = geocentric orbital eccentricity of the spacecraft(non-dimensional)

inc-geo (deg) = geocentric orbital inclination of the spacecraft in degrees

argper-geo (deg) = geocentric argument of perigee of the spacecraft in degrees

raan-geo (deg) = geocentric right ascension of the ascending node of the spacecraft in degrees

tanom-geo (deg) = geocentric true anomaly of the spacecraft in degrees

The `e2m_helio.csv` disk file contains the following information.

time (days) = simulation time since TCM maneuver in days

rs2sc-x (au) = x-component of the spacecraft's heliocentric position vector in astronomical units

rs2sc-y (au) = y-component of the spacecraft's heliocentric position vector in astronomical units

rs2sc-z (au) = z-component of the spacecraft's heliocentric position vector in astronomical units

rs2sc-mag (au) = the spacecraft's heliocentric radius in astronomical units

vs2sc-y (km/sec) = y-component of the spacecraft's heliocentric velocity vector in kilometers per second

vs2sc-z (km/sec) = z-component of the spacecraft's heliocentric position vector in kilometers per second

vs2sc-mag (km/sec) = the spacecraft's heliocentric speed in kilometers per second

rs2e-x (au) = x-component of Earth's heliocentric position vector in astronomical units

rs2e-y (au) = y-component of Earth's heliocentric position vector in astronomical units

rs2e-z (au) = z-component of Earth's heliocentric position vector in astronomical units

rs2e-mag (au) = Earth heliocentric radius in astronomical units

rs2m-x (au) = x-component of Mars heliocentric position vector in astronomical units

rs2m-y (au) = y-component of Mars heliocentric position vector in astronomical units

rs2m-z (au) = z-component of Mars heliocentric position vector in astronomical units

rs2m-mag (au) = Mars heliocentric radius in astronomical units

sma-heo (au) = heliocentric semimajor axis of the spacecraft in astronomical units

ecc-heo = heliocentric orbital eccentricity of the spacecraft(non-dimensional)

inc-heo (deg) = heliocentric orbital inclination of the spacecraft in degrees

argper-heo (deg) = heliocentric argument of perigee of the spacecraft in degrees

raan-heo (deg) = heliocentric right ascension of the ascending node of the spacecraft in degrees

tanom-heo (deg) = heliocentric true anomaly of the spacecraft in degrees

The `e2m_areo.csv` disk file contains the following information.

time (days) = simulation time since exit from the Earth's SOI in days

rm2sc-x (km) = x-component of the spacecraft's areocentric position vector in kilometers

rm2sc-y (km) = y-component of the spacecraft's areocentric position vector in kilometers

rm2sc-z (km) = z-component of the spacecraft's areocentric position vector in kilometers

rm2sc-mag (km) = the spacecraft's areocentric radius in kilometers

vm2sc-y (km/sec) = y-component of the spacecraft's areocentric velocity vector in kilometers per second

vm2sc-y (km/sec) = y-component of the spacecraft's areocentric velocity vector in kilometers per second

vm2sc-z (km/sec) = z-component of the spacecraft's areocentric position vector in kilometers per second

vm2sc-mag (km/sec) = the spacecraft's areocentric speed in kilometers per second

sma-areo (km) = areocentric semimajor axis of the spacecraft in kilometers

ecc-areo = areocentric orbital eccentricity of the spacecraft(non-dimensional)

inc-areo (deg) = areocentric orbital inclination of the spacecraft in degrees

argper-areo (deg) = areocentric argument of perigee of the spacecraft in degrees

raan-areo (deg) = areocentric right ascension of the ascending node of the spacecraft in degrees

tanom-areo (deg) = areocentric true anomaly of the spacecraft in degrees

rp-aero (km) = areocentric periapsis radius in kilometers

b dot t = dot product of the B-plane b and t vectors

b dot r = dot product of the B-plane b and r vectors

The geocentric and heliocentric coordinates of the spacecraft are with respect to the Earth mean equator and equinox of J2000 (EME2000) coordinate system. The areocentric coordinates of the spacecraft are with respect to the areocentric (Mars-centered) mean equator and IAU node of epoch coordinate system.

"Update to Mars Coordinate Frame Definitions", R. A. Mase, JPL IOM 312.B/015-99, 15 July 1999.

"The Planetary and Lunar Ephemeris DE 421", W. M. Folkner, J. G. Williams, and D. H. Boggs, JPL IOM 343R-08-003, 31 March 2008.

"Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000", *Celestial Mechanics and Dynamical Astronomy*, **82**: 83-110, 2002.

"IERS Conventions (2003)", IERS Technical Note 32, November 2003.

"Planetary Constants and Models", R. Vaughan, JPL D-12947, December 1995.

"Preliminary Mars Planetary Constants and Models for Mars Sample Return", D. Lyons, JPL IOM 312/99.DTL-1, 20 January 1999.

"Interplanetary Mission Design Handbook, Volume 1, Part 2", JPL Publication 82-43, September 15, 1983.

"An *Introduction to the Mathematics and Methods of Astrodynamics*, Revised Edition", Richard H. Battin, AIAA Education Series, 1999.

"A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem", Robert H. Gooding, *Celestial Mechanics and Dynamical Astronomy* **48:** 145-165, 1990.

CMATO 13 – Interplanetary Gravity-Assist Trajectory Optimization

This *CMATO* application is a Fortran computer program named `flyby_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the classic one impulse flyby and two-impulse rendezvous, *ballistic* gravity-assist interplanetary trajectory optimization problems. The software attempts to minimize the launch delta-v, the arrival delta-v or the total mission delta-v. The type of trajectory optimization is specified by the user. The destination object for the simulation can also be a comet or asteroid.

The important features of this scientific simulation are as follows:

- one-impulse, *patched-conic* interplanetary destination flyby trajectory modeling
- two-impulse, *patched-conic* interplanetary *rendezvous* trajectory modeling
- n-body heliocentric, inertial cartesian equations of motion
- elliptical, non-coplanar planetary, asteroid and comet orbits
- JPL DE421 planetary ephemeris model

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `flyby_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's problem and perform the optimization using a sparse nonlinear programming (NLP) method. The `flyby_sos` software allows the user to select the type of collocation method and other important algorithm control parameters.

Input file format and contents

The `flyby_sos` computer program is "data-driven" by a simple text file created by the user. The contents of this file include such things as initial guesses, the discretization method, and algorithm control parameters. The following is a typical input file used by this computer program. This example is an Earth-to-Mars trajectory with a gravity-assist from Venus. The performance index for this example is the total delta-v for the mission. Please note for a flyby trajectory input file (`trajectory type = 1`), the only optimization option is `1 = minimize launch delta-v`.

In the following discussion the actual input file contents are in *courier* font and all explanations are in times font. Please note the fundamental time argument in this computer program is Barycentric Dynamical Time (TDB).

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is <u>not</u> case sensitive but must be spelled correctly.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
******************************************************
** gravity-assist interplanetary trajectory optimization
** patched-conic heliocentric motion
** Earth-to-Venus-to-Mars - evm.in
** May 10, 2011
******************************************************
```

The first program input is an integer that defines the type of delta-v optimization. Please note that option 4, `no optimization` simply solves the orbital two-point boundary value problem subject to the user-defined flyby altitude constraint.

```
******************
* simulation type *
******************
 1 = minimize launch delta-v
 2 = minimize arrival delta-v
 3 = minimize total delta-v
 4 = no optimization
---------------------------------
3
```

The next input is an integer that tells the simulation what type of trajectory to model.

```
trajectory type (1 = flyby at destination, 2 = rendezvous)
2
```

The next three inputs are the user's initial guess for the launch calendar date. Be sure to include all four digits of the calendar year.

```
launch calendar date initial guess (month, day, year)
2,1,2009
```

The software allows the user to specify an initial guess for the launch, flyby and arrival calendar dates, and lower and upper bounds on the actual launch, flyby and arrival calendar dates found during the

optimization process. For any guess for launch time t_L and user-defined launch time lower and upper bounds Δt_l and Δt_u, the launch time t is constrained as

$$t_L - \Delta t_l \le t \le t_L + \Delta t_u$$

Likewise, for any guess for arrival time t_A and user-defined arrival time bounds Δt_l and Δt_u, the arrival time t is constrained as follows

$$t_A - \Delta t_l \le t \le t_A + \Delta t_u$$

For fixed launch, flyby or arrival times, the lower and upper bounds are set to 0.

The next two inputs are the lower and upper bounds for the launch calendar date search interval. These values should be input in days.

```
launch date search boundary (days)
-10, +10
```

The next three inputs are the user's initial guess for the flyby calendar date. Be sure to include all four digits of the calendar year.

```
flyby calendar date initial guess (month, day, year)
6,1,2009
```

The next two inputs are the lower and upper bounds for the flyby calendar date search interval. These values should be input in days.

```
flyby date search boundary (days)
-60, +60
```

The next three inputs are the user's initial guess for the arrival calendar date. Be sure to include all four digits of the calendar year.

```
arrival calendar date initial guess (month, day, year)
1,17,2010
```

The next two inputs are the lower and upper bounds for the arrival calendar date search interval. These values should be input in days.

```
arrival date search boundary (days)
-30, +30
```

The next two program inputs are integers that specify the launch and flyby planets.

```
*****************
* launch planet *
*****************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
----------
3
```

```
****************
* flyby planet *
****************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
----------
2
```

This next input defines the altitude constraint during the gravity assist. This altitude is with respect to a spherical flyby planet.

```
**************************
flyby altitude (kilometers)
**************************
500.0
```

This integer specifies the arrival planet or asteroid/comet.

```
*************************
* arrival celestial body *
*************************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
 0 = asteroid/comet
-------------------
4
```

The next series of inputs include the name and classical orbital elements of a comet or asteroid (arrival celestial body = 0). Please note that the angular orbital elements must be specified with respect to a heliocentric, Earth mean ecliptic and equinox of J2000 coordinate system. The calendar date of perihelion passage should be with respect to the TDB time system. Please consult *Appendix F – Computing an Asteroid or Comet Ephemeris* for additional information.

```
*********************************
* asteroid/comet orbital elements *
* (heliocentric, ecliptic J2000)  *
*********************************

asteroid/comet name
Tempel 1

calendar date of perihelion passage (month, day, year)
7, 5.3153, 2005

perihelion distance (au)
1.506167

orbital eccentricity (nd)
0.517491
```

```
orbital inclination (degrees)
10.5301

argument of perihelion (degrees)
178.8390

longitude of the ascending node (degrees)
68.9734
```

This next input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by the *Sparse Optimization Suite*. Options 2 and 3 allow the user to specify either the number of nodes (option 2) or time step size of the data file (option 3).

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this next input defines either the number of data points (option 2) or the time step size of the data output in the solution file (option 3).

```
number of user-defined nodes or print step size in solution data file
0.1
```

The name of the solution data file is defined in this next line.

```
name of solution output file
evm.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next integer defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the differential equation algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
****************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the Sparse Optimization Suite user's manual. To ignore this special output control, input the simple character string no.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

Optimal control solution

The `flyby_sos` computer program will create a comma-separated-variable (csv) solution file and a screen display of important trajectory characteristics. The csv file contains the state vector and classical orbital elements of the spacecraft's heliocentric trajectory. The `flyby_sos` software will also create a comma-separated-variable file named `planets.csv`. This file contains the heliocentric, ecliptic state vectors of all three celestial bodies.

The following is the screen display created by the simulation for this example.

```
 program flyby_sos
 ==================

 input file ==> evm.in

 rendezvous trajectory

 minimize total delta-v

 LAUNCH CONDITIONS
 =================

 calendar date            01/26/2009

 TDB time                 22:53:42.19

 TDB julian date          2454858.45396054

 heliocentric delta-v vector and magnitude
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

 launch delta-vx     3598.61873453787      meters/second
 launch delta-vy    -107.914756217745      meters/second
 launch delta-vz    -4086.58387702196      meters/second
launch delta-v      5446.27121754988      meters/second

 launch hyperbola characteristics
 (Earth mean equator and equinox of J2000)
 ------------------------------------------

 right ascension     22.9866977993573      degrees

 declination        -44.1317082595752      degrees

 C3                  29.6618701751123      (km/sec)**2

 heliocentric orbital elements and state vector of the departure planet
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

      sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.1000894396D+01    0.1748957901D-01    0.1517825095D-02    0.1876680899D+03

    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.2765666604D+03    0.2277150922D+02    0.2104395991D+03    0.3657470344D+03

      rx (km)             ry (km)             rz (km)             rmag (km)
 -.8866639772D+08    0.1176375427D+09    -.1977073770D+04    0.1473102900D+09

      vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 -.2428294068D+02    -.1805025766D+02    -.6937433411D-03    0.3025678453D+02
```

heliocentric orbital elements and state vector of the spacecraft
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.8633495757D+00	0.1583750760D+00	0.8469932720D+01	0.3365010516D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.3070010956D+03	0.2035041692D+03	0.1800052208D+03	0.2930073042D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.8866639772D+08	0.1176375427D+09	-.1977073770D+04	0.1473102900D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.2068432194D+02	-.1815817242D+02	-.4087277620D+01	0.2782563994D+02

FLYBY CONDITIONS
==========

calendar date	06/02/2009	
TDB time	08:21:01.61	
TDB julian date	2454984.84793527	
incoming v-infinity	7461.87134854987	meters/second
outgoing v-infinity	7461.87134856885	meters/second
flyby altitude	499.999999995290	kilometers
maximum turn angle	58.7932373404984	degrees
actual turn angle	56.2034020701081	degrees
true anomaly at infinity	118.101701035054	degrees

orbital elements of flyby hyperbola at periapsis
(planet-centered, Earth ecliptic and equinox of J2000)
---

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.5834435633D+04	0.2122970654D+01	0.7257834718D+02	0.3896208002D+02

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.2372825935D+03	0.0000000000D+00	0.3896208002D+02	0.0000000000D+00

heliocentric delta-v vector and magnitude
(Earth mean ecliptic and equinox of J2000)
---

flyby delta-vx	1840.87465555144	meters/second
flyby delta-vy	5314.10733376293	meters/second
flyby delta-vz	-4217.51542566270	meters/second
flyby delta-v	7029.65095989178	meters/second
maximum delta-v	7326.58026641160	meters/second

heliocentric orbital elements of the spacecraft after flyby
(Earth mean ecliptic and equinox of J2000)
---

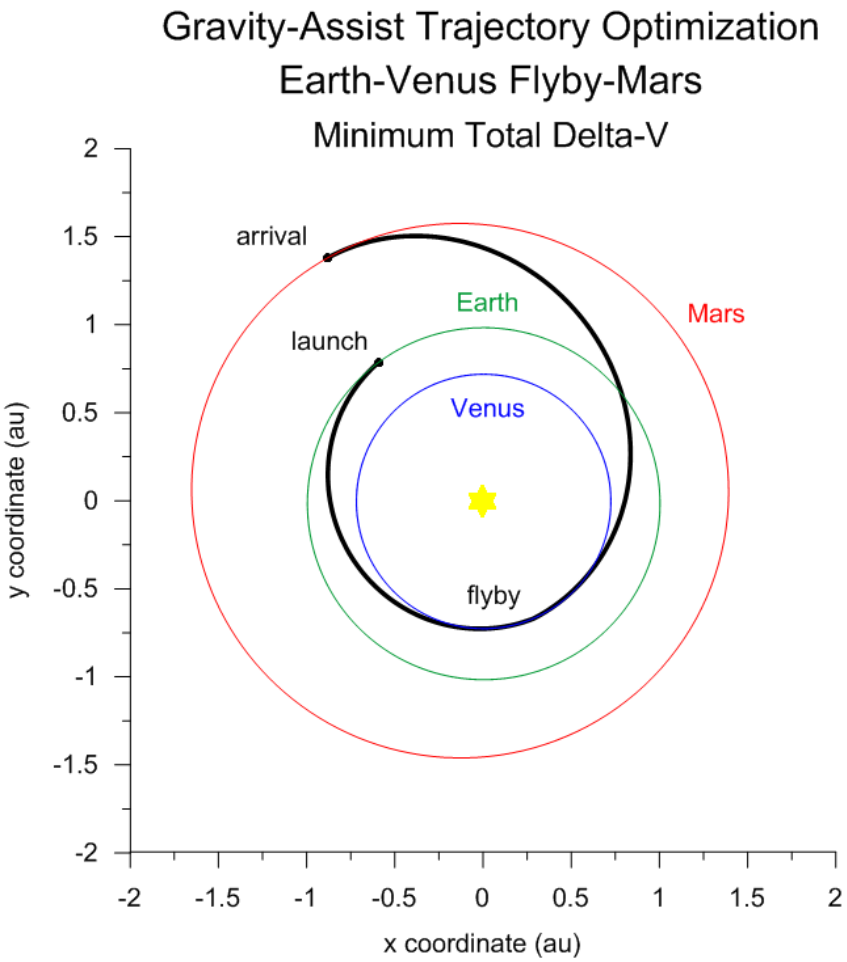
sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1182404983D+01	0.3907551658D+00	0.2468726353D+01	0.3205429221D+03

```
   raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.3483870514D+03    0.3443052834D+03    0.3048482055D+03    0.4696213189D+03
```

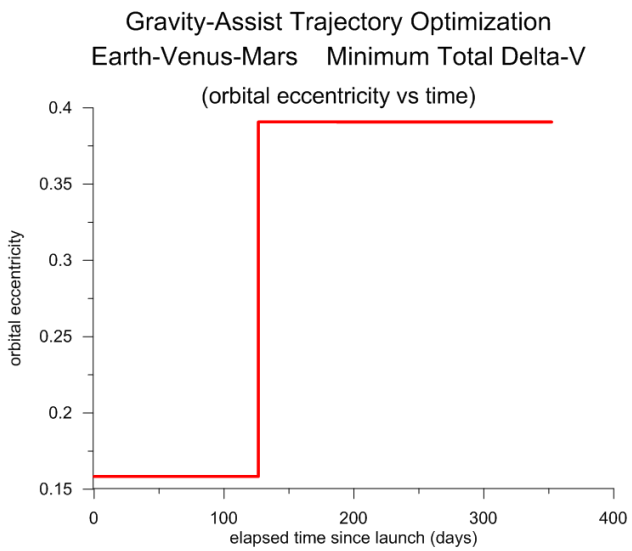
ARRIVAL CONDITIONS

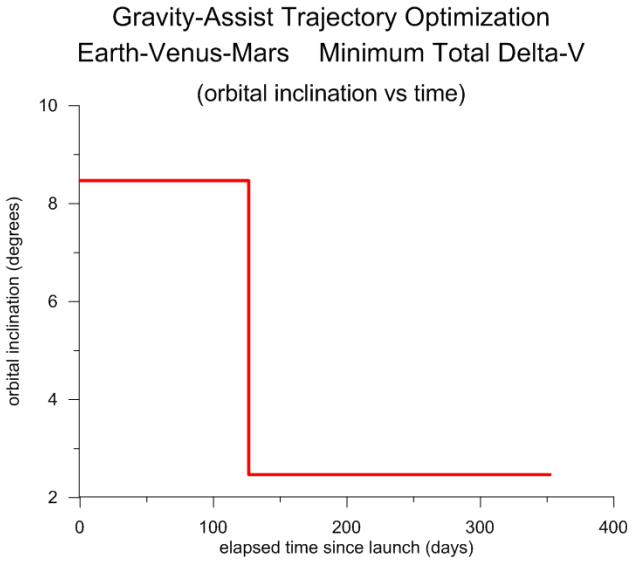
```
==================

calendar date            01/14/2010

TDB time                 04:03:00.82

TDB julian date          2455210.66875952

heliocentric delta-v vector and magnitude
(Earth mean ecliptic and equinox of J2000)
----------------------------------------

arrival delta-vx    3466.49432672797      meters/second
arrival delta-vy    2288.97943583604      meters/second
arrival delta-vz   -755.089179469042      meters/second

arrival delta-v     4222.10485941179      meters/second

C3                  17.8261694438687      km**2/sec**2

heliocentric orbital elements and state vector of the spacecraft
(Earth mean ecliptic and equinox of J2000)
----------------------------------------

     sma (au)         eccentricity       inclination (deg)     argper (deg)
 0.1523705057D+01    0.9333969433D-01    0.1848903776D+01    0.2865565867D+03

   raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.4952542122D+02    0.1465550304D+03    0.7311161711D+02    0.6869890110D+03

     rx (km)             ry (km)             rz (km)             rmag (km)
 -.1320620326D+09    0.2062715942D+09    0.7565034318D+07    0.2450419980D+09

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 -.1948822015D+02    -.1100506230D+02    0.2479500494D+00    0.2238221616D+02

heliocentric orbital elements and state vector of the destination body
(Earth mean ecliptic and equinox of J2000)
----------------------------------------

     sma (au)         eccentricity       inclination (deg)     argper (deg)
 0.1523705057D+01    0.9333969435D-01    0.1848903776D+01    0.2865565867D+03

   raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.4952542122D+02    0.1465550305D+03    0.7311161711D+02    0.6869890110D+03


     rx (km)             ry (km)             rz (km)             rmag (km)
 -.1320620326D+09    0.2062715942D+09    0.7565034318D+07    0.2450419980D+09

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 -.1948822015D+02    -.1100506230D+02    0.2479500494D+00    0.2238221616D+02
```

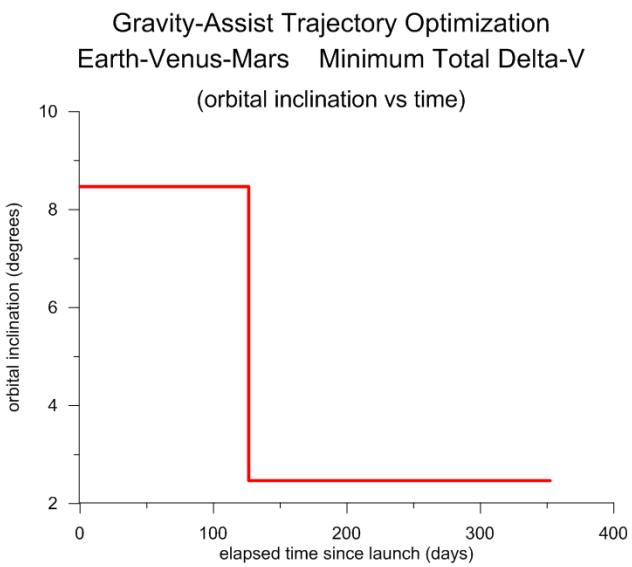
The following is the heliocentric transfer trajectory for this example. It is an inertial view of the trajectory and planetary orbits from the north pole of the ecliptic looking down on the ecliptic plane. The transfer trajectory is black and the scales are in astronomical units.

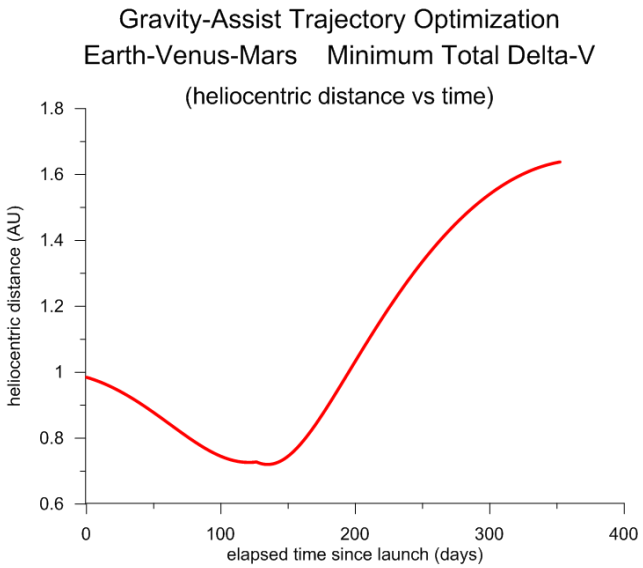


Several trajectory characteristics for this mission are shown in the following plots. These plots illustrate the effect of the gravity assist on the spacecraft's heliocentric classical orbital elements.

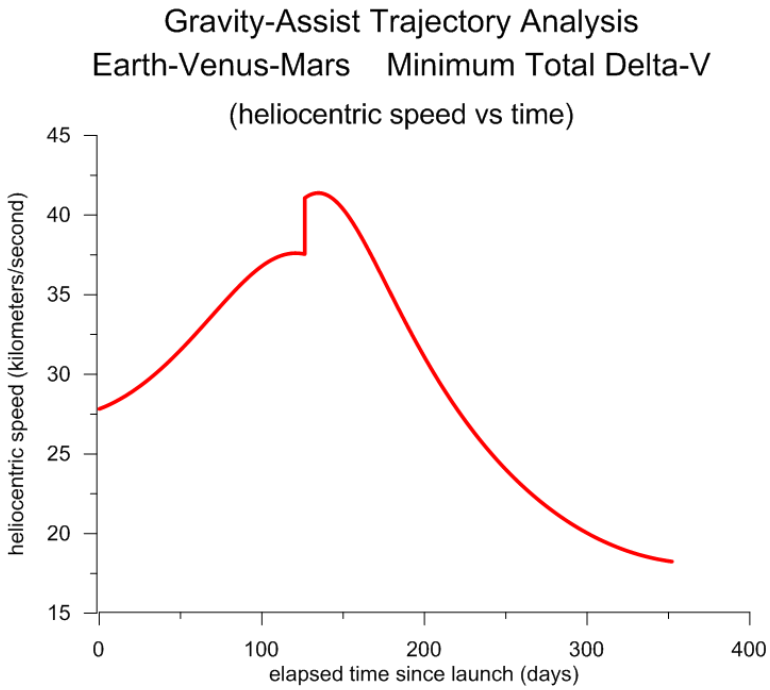








This final plot illustrates the change in heliocentric speed produced by the flyby.



Problem setup

This section provides additional details about the software implementation. For good scaling, the time unit used in all internal calculations is days, position is expressed in astronomical units, and the velocity and delta-v units are astronomical units per day.

This classic trajectory optimization problem is formulated using two mission phases. The first phase is the interplanetary trajectory from launch to the gravity assist at the flyby planet and the second phase is the transfer trajectory from the gravity assist to arrival at the destination planet.

Launch, flyby and arrival time bounds

The software allows the user to specify an initial guess for the launch calendar date and the durations of the first and second legs of the interplanetary transfer trajectory. The user can also provide lower and upper bounds on the actual launch, flyby and arrival dates found during the optimization process.

For any guess for launch time t_L and user-defined launch time lower and upper bounds Δt_l and Δt_u, the launch time t is constrained as

$$t_L - \Delta t_l \le t \le t_L + \Delta t_u$$

Likewise, for any guess for flyby time t_{FB} and user-defined bounds Δt_l and Δt_u, the flyby time t is constrained as follows

$$t_{FB} - \Delta t_l \le t \le t_{FB} + \Delta t_u$$

Finally, for any guess for arrival time t_A and user-defined arrival time bounds Δt_l and Δt_u, the arrival time t is constrained as follows:

$$t_A - \Delta t_l \le t \le t_A + \Delta t_u$$

For fixed launch, flyby and/or arrival times, the lower and upper bounds are set to 0.

Performance index – minimize delta-v

The objective function or performance index *J* for this simulation is one of three possible delta-v's. For this classic trajectory optimization problem, this index is simply $J = \Delta V$ where ΔV is either the launch delta-v, arrival delta-v or the total delta-v. The value of the `maxmin` indicator tells the *Sparse Optimization Suite* whether the user is minimizing or maximizing the performance index. The type of delta-v optimization is selected by the user and the performance index is enforced at the end of phase 2.

Point functions – position and velocity vector "matching" at launch

For any launch time t_L the optimal solution for a rendezvous trajectory must satisfy the following state vector boundary conditions (equality constraints) at launch:

$$\mathbf{r}_{s/c}\left(t_L\right) - \mathbf{r}_p\left(t_L\right) = 0$$

$$\mathbf{v}_{s/c}\left(t_L\right) - \left\{\mathbf{v}_p\left(t_L\right) + \Delta\mathbf{v}\left(t_L\right)\right\} = 0$$

where $\mathbf{r}_{s/c}$ and $\mathbf{v}_{s/c}$ are the heliocentric, inertial position and velocity vectors of the spacecraft at the launch time t_L, $\mathbf{r}_p$ and $\mathbf{v}_p$ are the heliocentric, inertial position and velocity vectors of the departure planet at the launch time, and $\Delta\mathbf{v}$ is the *impulsive* heliocentric delta-v vector required at launch. These point constraints are enforced at the beginning of the first phase.

Point functions – v-infinity and flyby altitude "matching" at flyby

Point functions are required to enforce the v-infinity and altitude "match" at the flyby. These two scalar equality constraints are given by

$$\left|\mathbf{v}_{\infty}^{-}\right| - \left|\mathbf{v}_{\infty}^{+}\right| = 0 \qquad h_{fb} - h_t = 0$$

where h_{fb} is the actual flyby altitude and h_t is the "targeted" or user-defined flyby altitude. The components of the incoming v-infinity velocity vector $\mathbf{v}_{\infty}^{-}$ are "saved" in parameters at the end of phase 1 and the outgoing v-infinity velocity vector $\mathbf{v}_{\infty}^{+}$ and flyby altitude are computed at the beginning of phase 2. These constraints are enforced at the beginning of phase 2 using point functions that consist of the scalar difference between the v-infinity vectors and altitude.

Linkage conditions – time and position vector across the flyby

Because the interplanetary transfer trajectory needs to be contiguous in time and heliocentric position, the time and heliocentric position vector of the spacecraft must be "linked" across the flyby boundary. This is accomplished using the `linkst` subroutine that is part of the *Sparse Optimization Suite*. This linkage is enforced at the beginning of phase 2.

Point functions – position and velocity vector "matching" at arrival

For any arrival time t_A the optimal solution for a rendezvous trajectory must satisfy the following state vector boundary conditions at arrival:

$$\mathbf{r}_{s/c}\left(t_A\right) - \mathbf{r}_p\left(t_A\right) = 0$$

$$\mathbf{v}_{s/c}\left(t_A\right) - \left\{\mathbf{v}_p\left(t_A\right) + \Delta\mathbf{v}\left(t_A\right)\right\} = 0$$

where $\mathbf{r}_{s/c}$ and $\mathbf{v}_{s/c}$ are the heliocentric, inertial position and velocity vectors of the spacecraft at the arrival time t_A, and $\mathbf{r}_p$ and $\mathbf{v}_p$ are the heliocentric, inertial position and velocity vectors of the destination planet at the arrival time, and $\Delta\mathbf{v}$ is the *impulsive* heliocentric delta-v vector required at arrival.. This system of launch and arrival state vector equality constraints ensures a rendezvous mission. These point constraints are enforced at the end of phase 2. For a destination flyby mission, the velocity vector point functions at arrival are not enforced.

The components of the launch and arrival impulsive delta-v's are the control variables used by the Sparse Optimization Suite to solve this trajectory problem.

Creating an initial guess

An initial guess for the *Sparse Optimization Suite* is created by solving the Lambert two-point boundary-value problem (TPBVP) for each leg of the interplanetary transfer trajectory using the launch calendar date and transfer time initial guesses provided by the user. The computational steps which create the initial guess for the spacecraft state, and the launch and arrival delta-v are as follows:

1) compute the state vector of the departure planet at the departure date initial guess
2) compute the state vector of the flyby planet at the flyby date initial guess
3) compute the state vector of the arrival planet at the arrival date initial guess

4) solve Lambert's problem for the departure-to-flyby leg and determine the initial velocity vector of this leg

5) compute the launch delta-v vector from the launch planet's velocity vector and the initial velocity vector of the first leg of the transfer trajectory

6) solve Lambert's problem for the second heliocentric leg and determine the initial and final velocity vectors of the second leg

7) compute the arrival delta-v vector from the arrival planet's velocity vector and the final velocity vector of the second leg of the transfer trajectory

8) estimate the flyby incoming v-infinity vector from the flyby planet's velocity vector and the final velocity vector of the first leg of the transfer trajectory

The heliocentric state vector at the end of each leg of the interplanetary cruise part of the mission is computed using Sheppard's *two-body* orbit propagation algorithm. The derivation of this algorithm is described in "Universal Keplerian State Transition Matrix", *Celestial Mechanics*, **35** (1985) 129-144. This numerical method uses a combination of universal variables and continued fractions to analytically propagate two-body orbits. *Appendix I – Numerical Solutions of Lambert's Problem* contains additional information about Lambert's problem and its numerical solution.

The Lambert solution initializes the `flyby_sos` software using the operator's initial guess for launch and arrival dates. The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 5` within the `odeinp` subroutine for this aerospace trajectory optimization problem. These program options create an initial guess from the numerical integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the software to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 5` program option tells the software to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

The algorithm used in this computer program to solve the two-body Lambert problem is based on the method described in "A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem" by R. H. Gooding, *Celestial Mechanics and Dynamical Astronomy* **48:** 145-165, 1990. This iterative solution is valid for elliptic, parabolic and hyperbolic transfer orbits which may be either posigrade or retrograde, and involve one or more revolutions about the central body.

Technical discussion

The $\Delta\mathbf{V}$ required at launch and arrival are simply the differences between the velocity on the transfer trajectory determined by the solution of Lambert's problem and the heliocentric velocities of the two planets. If we treat each planet as a *point mass* and assume *impulsive* maneuvers, the *planet-centered* magnitude and direction of the required maneuvers are given by the two vector equations:

$$\Delta\mathbf{V}_L = \mathbf{V}_{T_L} - \mathbf{V}_{P_L}$$

$$\Delta\mathbf{V}_A = \mathbf{V}_{P_A} - \mathbf{V}_{T_A}$$

where

$\mathbf{V}_{T_L}$ = heliocentric velocity vector of the transfer trajectory at launch

$\mathbf{V}_{T_A}$ = heliocentric velocity vector of the transfer trajectory at arrival

$\mathbf{V}_{P_L}$ = heliocentric velocity vector of the launch planet

$\mathbf{V}_{P_A}$ = heliocentric velocity vector of the arrival planet

The scalar magnitude of each maneuver is also called the "hyperbolic excess velocity" or V_∞ at launch and arrival. The hyperbolic excess velocity is the speed of the spacecraft relative to each planet at an *infinite* distance from the planet. Furthermore, the *energy* or C_3 at launch or arrival is equal to V_∞^2 for the respective maneuver. C_3 is also equal to twice the specific (per unit mass) orbital energy.

The orientation of the departure hyperbola is specified in terms of the right ascension and declination of the outgoing asymptote. These coordinates can be calculated using the individual components of the V_∞ velocity vector. For example, the right ascension of the asymptote is determined from $\alpha = \tan^{-1}\left(\Delta V_y, \Delta V_z\right)$ and the geocentric declination of the asymptote is given by $\delta = 90^0 - \cos^{-1}\left(\Delta \hat{V}_z\right)$ where $\Delta \hat{V}_x$, $\Delta \hat{V}_y$ and $\Delta \hat{V}_z$ are the x, y and z components of the unit ΔV vector. The right ascension is computed using a four-quadrant inverse tangent function.

In this computer program the heliocentric planetary coordinates and therefore the $\Delta\mathbf{V}$ vectors are computed in the Earth mean ecliptic and equinox of J2000 coordinate system. The geometry and transformations relative to this system are given in *Appendix G – Aerospace Trajectory Coordinates and Time Systems*.

Equations of motion

This computer program implements Cowell's form of the equations of motion as described in *Appendix C – Cartesian Equations of Motion*. All planetary perturbations except those due to the launch, flyby and arrival planets are included in the equations of motion. The equations described here are coded in the right-hand-side subroutine required by the *Sparse Optimization Suite*.

Gravity-assist flight mechanics

During a planetary flyby, the vector relationships between the incoming v-infinity vector $\mathbf{v}_\infty^-$, the outgoing v-infinity vector $\mathbf{v}_\infty^+$ and the two legs of the heliocentric transfer orbit that "patch" at the center of the flyby planet are as follows

$$\mathbf{V}_\infty^- = \mathbf{V}_{fb} - \mathbf{V}_{to_1}$$

$$\mathbf{V}_\infty^+ = \mathbf{V}_{to_2} - \mathbf{V}_{fb}$$

where

$\mathbf{V}_{fb}$ = heliocentric velocity vector of the flyby planet at the flyby date

$\mathbf{V}_{to_1}$ = heliocentric velocity vector of the first transfer orbit at the flyby date

$\mathbf{V}_{to_2}$ = heliocentric velocity vector of the second transfer orbit at the flyby date

The turn angle of the planet-centered trajectory during the flyby is determined from

$$\delta = 2\sin^{-1}\left(\frac{1}{1+r_p V_\infty^2/\mu}\right) = 2\theta_\infty - 180^\circ$$

where r_p is the periapsis radius of the flyby hyperbola, V_∞ is the magnitude of the incoming v-infinity vector, μ is the gravitational constant of the flyby planet and θ_∞ is the true anomaly at infinity.

The maximum turn angle possible during a gravity assist flyby occurs when the spacecraft just gazes the planet's surface. It is given by

$$\delta_{\max} = 2\sin^{-1}\left(\frac{1}{1+r_e V_\infty^2/\mu}\right)$$

where r_e is the radius of the flyby planet.

The semimajor axis and orbital eccentricity of the flyby hyperbola are given by

$$a = -\mu/\left|\mathbf{V}_\infty^-\right|^2 = -\mu/\left|\mathbf{V}_\infty^+\right|^2 \qquad e = -1/\cos\theta_\infty = 1 - \frac{r_p}{a} = 1 + \frac{r_p V_\infty^2}{\mu}$$

where θ_∞ is the true anomaly at infinity which is determined from $\theta_\infty = \cos^{-1}(-1/e)$.

The periapsis radius of the flyby hyperbola relative to a *spherical* planet is determined from the expression $r_p = a(1-e)$, and the flyby altitude is $h = r - r_e$ where *r* is the planet-centered radius at the flyby closest approach.

The heliocentric speed gained during the flyby and the heliocentric delta-v vector caused by the close encounter can be determined from the following two equations

$$\Delta V = 2V_\infty / e$$

$$\Delta\mathbf{V} = \mathbf{V}_h^- - \mathbf{V}_h^+$$

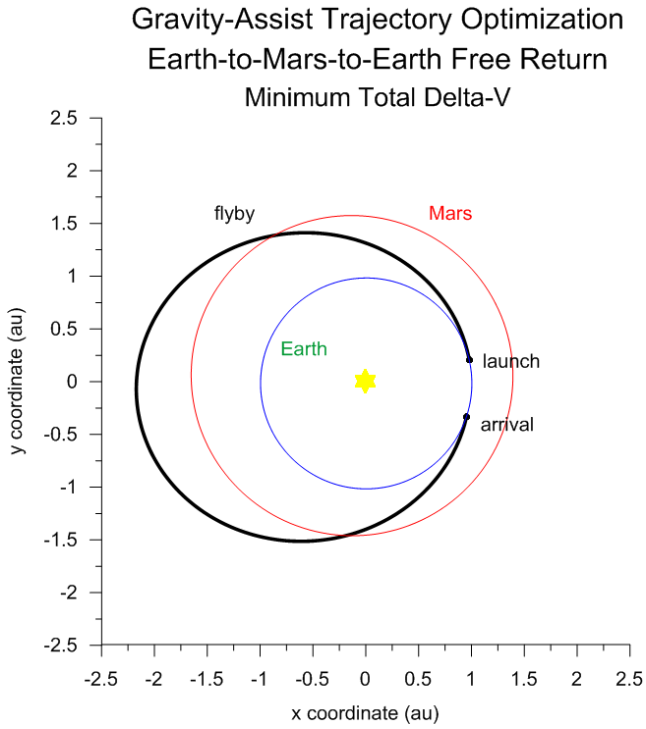
In the second equation, $\mathbf{V}_h^-$ is the heliocentric velocity vector of the spacecraft prior to the flyby and $\mathbf{V}_h^+$ is the heliocentric velocity vector after the flyby. The maximum heliocentric delta-v available from a gravity assist corresponds to a flyby at the planet's surface and is given by $\Delta V = \sqrt{\mu/r_e}$. This value is also equal to the "local circular velocity" at the flyby planet's surface.

Important Note

The binary ephemeris file provided with this computer program was created for use on Windows compatible computers. For other platforms, you will need to create or obtain binary files specific to that system. Information and computer programs for creating these files can be found at the JPL solar system FTP site located at ftp://ssd.jpl.nasa.gov/pub/eph/export/. This site provides ASCII data files and Fortran computer programs for creating a binary file.

Example free return trajectory

The following is the graphics display for a typical Earth free return trajectory computed with the `flyby_sos` software. It involves a flyby of Mars at an altitude of 15,000 kilometers.



Here's the input data file for this example.

```
*********************************************************
** gravity-assist interplanetary trajectory optimization
** patched-conic heliocentric motion
** Earth-to-Mars-to-Earth free return - freertn.in
** May 10, 2011
*********************************************************

******************
* simulation type *
******************
 1 = minimize launch delta-v
 2 = minimize arrival delta-v
 3 = minimize total delta-v
 4 = no optimization
--------------------
3

trajectory type (1 = flyby, 2 = rendezvous)
2

launch calendar date initial guess (month, day, year)
9,1,2007

launch date search boundary (days)
-60, +60

flyby calendar date initial guess (month, day, year)
2,1,2008

flyby search boundary (days)
-60, +60

arrival calendar date initial guess (month, day, year)
8,1,2009
```

```
arrival search boundary (days)
-60, +60

*****************
* launch planet *
*****************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
----------
3

****************
* flyby planet *
****************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
----------
4

***************************
flyby altitude (kilometers)
***************************
15000.0
************************
* arrival celestial body
************************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
 0 = asteroid/comet
--------------------
3

*********************************
* asteroid/comet orbital elements *
* (heliocentric, ecliptic J2000)  *
*********************************

asteroid/comet name
Tempel 1

calendar date of perihelion passage (month, day, year)
7, 5.3153, 2005

perihelion distance (au)
1.506167

orbital eccentricity (nd)
0.517491

orbital inclination (degrees)
10.5301

argument of perihelion (degrees)
178.8390
```

```
longitude of the ascending node (degrees)
68.9734

*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1

number of user-defined nodes or print step size in solution data file
0.1

name of solution output file
freertn.csv

*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1

relative error in the objective function (performance index)
1.0d-5

relative error in the solution of the differential equations
1.0d-7

maximum number of mesh refinement iterations
20

maximum number of function evaluations
50000

maximum number of algorithm iterations
1000

**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2

**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1

****************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
```

```
1

******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j1k0l0m0n0o0p0q0r0
```

Here's the `flyby_sos` program output for this example.

```
 program flyby_sos
 =================

 input file ==> freertn.in

 rendezvous trajectory

 minimize total delta-v

 LAUNCH CONDITIONS
 =================

 calendar date            10/05/2007

 TDB time                 12:42:35.23

 TDB julian date          2454379.02957438

 heliocentric delta-v vector and magnitude
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

 launch delta-vx     221.166730955122      meters/second
 launch delta-vy     5278.33645148211      meters/second
 launch delta-vz     1146.19793279266      meters/second

 launch delta-v      5405.87829303105      meters/second

 launch hyperbola characteristics
 (Earth mean equator and equinox of J2000)
 ------------------------------------------

 right ascension     87.1138001170666      degrees

 declination         35.6564739814873      degrees

 C3                  29.2235201190643      (km/sec)**2

 heliocentric orbital elements and state vector of the departure planet
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

      sma (au)          eccentricity       inclination (deg)     argper (deg)
  0.1000336997D+01    0.1639428025D-01    0.9372601345D-03    0.6282599731D+02

     raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
  0.3902031360D+02    0.2699458384D+03    0.3327718357D+03    0.3654415494D+03

      rx (km)             ry (km)             rz (km)             rmag (km)
  0.1464529146D+09    0.3057466560D+08    -.1119756613D+04    0.1496103819D+09

      vx (kps)            vy (kps)            vz (kps)            vmag (kps)
  -.6564532009D+01    0.2905486659D+02    0.4368704868D-03    0.2978721795D+02

 heliocentric orbital elements and state vector of the spacecraft
 (Earth mean ecliptic and equinox of J2000)
 ------------------------------------------

      sma (au)          eccentricity       inclination (deg)     argper (deg)
  0.1601934151D+01    0.3763117415D+00    0.1881511147D+01    0.3551442891D+03

     raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
  0.1180520330D+02    0.4842649861D+01    0.3599869389D+03    0.7405688092D+03
```

```
     rx (km)             ry (km)             rz (km)            rmag (km)
 0.1464529146D+09    0.3057466560D+08    -.1119756613D+04    0.1496103819D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 -.6343365279D+01    0.3433320304D+02    0.1146634803D+01    0.3493310586D+02
```

FLYBY CONDITIONS

```
calendar date            02/26/2008

TDB time                 02:07:08.51

TDB julian date          2454522.58829299

incoming v-infinity         7471.44164008788      meters/second

outgoing v-infinity         7471.44164008931      meters/second

flyby altitude              14999.9999998685      kilometers

maximum turn angle          21.2338575439034      degrees

actual turn angle           4.58880528217535      degrees

true anomaly at infinity    92.2944026410877      degrees

orbital elements of flyby hyperbola at periapsis
(planet-centered, Earth ecliptic and equinox of J2000)
-----------------------------------------------------

     sma (km)         eccentricity       inclination (deg)     argper (deg)
 -.7672239669D+03    0.2497865655D+02    0.1066540435D+03    0.9414755578D+02

    raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.2947666598D+03    0.0000000000D+00    0.9414755578D+02    0.0000000000D+00

heliocentric delta-v vector and magnitude
(Earth mean ecliptic and equinox of J2000)
------------------------------------------

flyby delta-vx     173.395276897405      meters/second
flyby delta-vy     32.3478335633444      meters/second
flyby delta-vz    -571.631099503925      meters/second

flyby delta-v      598.226059534885      meters/second

maximum delta-v    3550.73054049584      meters/second

heliocentric orbital elements of the spacecraft after flyby
(Earth mean ecliptic and equinox of J2000)
------------------------------------------

     sma (au)         eccentricity       inclination (deg)     argper (deg)
 0.1579749071D+01    0.3764913659D+00    0.2818739878D+01    0.2408255234D+02

    raan (deg)      true anomaly (deg)     arglat (deg)        period (days)
 0.3408594221D+03    0.1171542494D+03    0.1412368017D+03    0.7252380637D+03
```

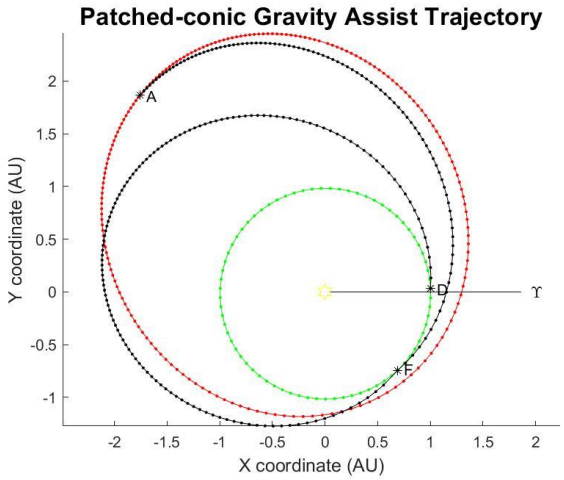
ARRIVAL CONDITIONS

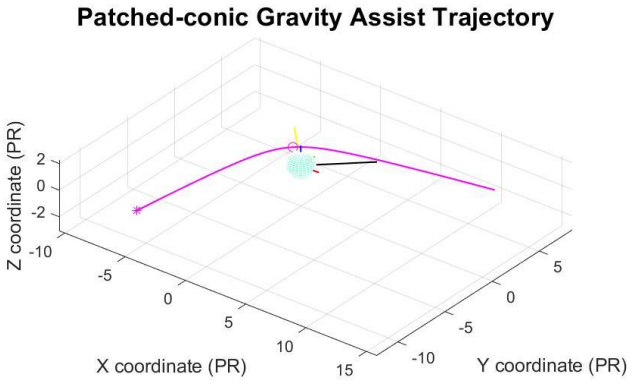
```
calendar date            09/03/2009

TDB time                 08:50:33.69

TDB julian date          2455077.86844547

heliocentric delta-v vector and magnitude
(Earth mean ecliptic and equinox of J2000)
------------------------------------------

arrival delta-vx   -1729.47403035797      meters/second
arrival delta-vy    5708.56371012704      meters/second
arrival delta-vz    1692.45664868647      meters/second
```

```
arrival delta-v     6200.25721740196      meters/second

C3                  38.4431895619451      km**2/sec**2

heliocentric orbital elements and state vector of the spacecraft
(Earth mean ecliptic and equinox of J2000)
----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.9991865067D+00    0.1717308235D-01    0.3095298195D-02    0.3011673501D+03

    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.1639272461D+03    0.2357679269D+03    0.1769352770D+03    0.3648112879D+03

     rx (km)             ry (km)             rz (km)             rmag (km)
 0.1425508125D+09    -.4946709095D+08    0.4358140058D+03    0.1508897850D+09

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 0.9275785768D+01    0.2802096131D+02    -.1593343549D-02    0.2951634254D+02

heliocentric orbital elements and state vector of the destination body
(Earth mean ecliptic and equinox of J2000)
----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.9991865067D+00    0.1717308235D-01    0.3095298195D-02    0.3011673501D+03

    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.1639272461D+03    0.2357679269D+03    0.1769352770D+03    0.3648112879D+03

     rx (km)             ry (km)             rz (km)             rmag (km)
 0.1425508125D+09    -.4946709095D+08    0.4358140058D+03    0.1508897850D+09

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 0.9275785768D+01    0.2802096131D+02    -.1593343549D-02    0.2951634254D+02
```

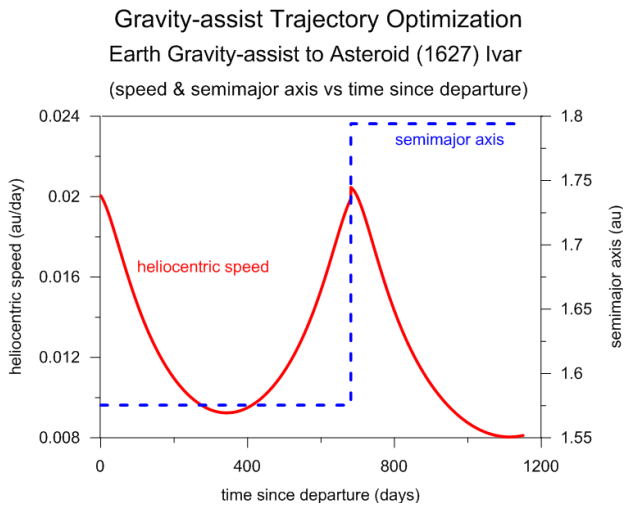
Example Earth gravity-assist trajectory to the asteroid (1627) Ivar

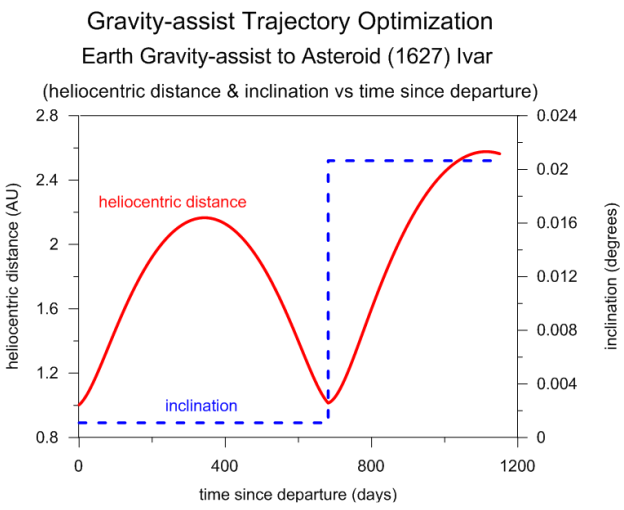
This section summarizes the trajectory characteristics for a patched-conic, gravity-assist interplanetary trajectory from the Earth to the asteroid (1627) Ivar. This example uses a single Earth gravity-assist during the trajectory optimization. The initial guess for this trajectory was extracted from the technical paper "The design of transfer trajectory for Ivar asteroid exploration mission", by D. Qiao, H. Cui, and P. Cui, *Acta Astronautica*, 65 (2009), pp. 1553-1560. Here are the heliocentric and flyby trajectories for this example. The Earth's orbit is green, the transfer trajectory trace is black and Ivar's orbit is red.





If we plot the behavior of the spacecraft's heliocentric trajectory, we can see how the gravity-assist affects the trajectory geometry as shown by the following two plots. It is easy to visualize the discontinuity in these elements at the gravity-assist event.





Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `flyby_sos` software. All output except the coordinates of the launch hyperbola is computed and displayed in a heliocentric, Earth mean ecliptic and equinox of J2000 coordinate system.

The simulation summary screen display contains the following information.

calendar date = calendar date of trajectory event

TDB time = Barycentric Dynamical Time of trajectory event

TDB julian date = TDB julian date of trajectory event

launch delta-vx = x-component of the impulsive velocity vector at launch in kilometers/second

launch delta-vy = y-component of the impulsive velocity vector at launch in kilometers/second

launch delta-vz = z-component of the impulsive velocity vector at launch in kilometers/second

launch delta-v = scalar magnitude of the impulsive delta-v at launch in kilometers/second

right ascension = right ascension of the launch hyperbola in degrees

declination = declination of the launch hyperbola in degrees

launch energy = launch energy in km/sec squared

sma (au) = semimajor axis in astronomical unit

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (days) = orbital period in days

rx (km) = x-component of spacecraft or celestial body position vector in kilometers

ry (km) = y-component of spacecraft or celestial body position vector in kilometers

rz (km) = z-component of spacecraft or celestial body position vector in kilometers

rmag (km) = heliocentric radius magnitude of spacecraft or celestial body in kilometers

vx (kps) = x-component of spacecraft or celestial body velocity vector in kilometers per second

vy (kps) = y-component of spacecraft or celestial body velocity vector in kilometers per second

vz (kps) = z-component of spacecraft or celestial body velocity vector in kilometers per second

vmag (kps) = heliocentric velocity magnitude of spacecraft or celestial body in kilometers per second

incoming v-infinity = incoming “speed at infinity” in kilometers per second

outgoing v-infinity = outgoing “speed at infinity” in kilometers per second

flyby altitude = altitude of the flyby in kilometers

maximum turn angle = the turn angle if the spacecraft grazed the flyby planet’s surface in degrees

actual turn angle = actual turn angle produced by the flyby in degrees

true anomaly at infinity = true anomaly of the spacecraft at an infinite distance from the flyby planet

flyby delta-vx = x-component of the heliocentric impulsive velocity vector due to the flyby in kilometers/second

flyby delta-vy = y-component of the heliocentric impulsive velocity vector due to the flyby in kilometers/second

flyby delta-vz = z-component of the heliocentric impulsive velocity vector due to the flyby in kilometers/second

flyby delta-v = scalar magnitude of the heliocentric impulsive delta-v due to the flyby in kilometers/second

maximum delta-v = the maximum possible heliocentric delta-v due to the flyby in kilometers/second

arrival delta-vx = x-component of the impulsive velocity vector at arrival in kilometers/second

arrival delta-vy = y-component of the impulsive velocity vector at arrival in kilometers/second

arrival delta-vz = z-component of the impulsive velocity vector at arrival in kilometers/second

arrival delta-v = scalar magnitude of the impulsive delta-v at arrival in kilometers/seconds

arrival energy = arrival energy in km/sec squared

The comma-separated-variable disk file is created by the *SOS* `odeprt` subroutine and contains the following information.

time (days) = simulation time since launch in days

rx (au) = x-component of spacecraft position vector in astronomical units

ry (au) = y-component of spacecraft position vector in astronomical units

rz (au) = z-component of spacecraft position vector in astronomical units

rmag (au) = heliocentric radius magnitude of spacecraft in astronomical units

vx (km/sec) = x-component of spacecraft velocity vector in kilometers per second

vy (km/sec) = y-component of spacecraft velocity vector in kilometers per second

vz (km/sec) = z-component of spacecraft velocity vector in kilometers per second

vmag (km/sec) = heliocentric velocity of spacecraft in kilometers per second

semimajor axis (au) = semimajor axis in astronomical units

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

arg of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (days) = orbital period in days

The `planets.csv` file contains the following information.

time (days) = simulation time since launch in days

rp1-x (au) = x-component of the launch planet position vector in astronomical units
rp1-y (au) = y-component of the launch planet position vector in astronomical units
rp1-z (au) = z-component of the launch planet position vector in astronomical units

rp2-x (au) = x-component of the destination body position vector in astronomical units
rp2-y (au) = y-component of the destination body position vector in astronomical units
rp2-z (au) = z-component of the destination body position vector in astronomical units

"Interplanetary Mission Design Handbook, Volume 1, Part 2", JPL Publication 82-43, September 15, 1983.

"Error Analysis of Multiple Planet Trajectories", F. M. Sturms, Jr., JPL Space Programs Summary, No. 37-27, Vol. IV.

"Optimal Interplanetary Orbit Transfers by Direct Transcription", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 42, No. 3, July-September 1994, pp. 247-268.

"Design and Optimization of Interplanetary Spacecraft Trajectories", Thomas T. McConaghy, PhD thesis, Purdue University, December 2004.

"A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem" by R. H. Gooding, *Celestial Mechanics and Dynamical Astronomy* **48:** 145-165, 1990.

CMATO 14 – Low-Thrust Interplanetary Trajectory Optimization

This *CMATO* application is a Fortran computer program named `ilt_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the low-thrust interplanetary flyby and rendezvous trajectory optimization problems. The software attempts to either maximize the final spacecraft mass or minimize the transfer time. The type of optimization is selected by the user. The interplanetary trajectory is modeled as a patched-conic, n-body system.

The important features of this scientific simulation are as follows:

- maximum payload or minimum transfer time optimization
- low-thrust, *patched-n-body* interplanetary *flyby* and *rendezvous* trajectory modeling
- user-defined bounds for throttle setting
- modified equinoctial orbital element equations of motion
- chemical or solar electric propulsion (SEP) models
- JPL DE421 planetary ephemeris model

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, "Practical Methods for Optimal Control and Estimation Using Nonlinear Programming" by John. T. Betts, SIAM, 2010.

The `ilt_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's problem and perform the optimization using a sparse nonlinear programming method. The software allows the user to select the type of initial guess, collocation method and other important algorithm control parameters.

Input file format and contents

The `ilt_sos` software is "data-driven" by a user-created text file. The following is a typical input file used by this computer program. This example is a maximum payload Earth-to-Mars rendezvous trajectory with an initial launch C_3 of 4.625 km^2/sec^2. Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is <u>not</u> case sensitive but must be spelled correctly.

In the following discussion the actual input file contents are in *courier* font and all explanations are in times font. Please note that the fundamental time argument in this computer program is Barycentric Dynamical Time (TDB). Furthermore, the fundamental coordinate system is the Earth mean ecliptic and equinox of J2000.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
************************************************
** low-thrust interplanetary trajectory optimization
** patched-conic, n-body heliocentric motion
** input file for ilt_sos - e2m1_max_payload.in
** Earth-to-Mars max payload – March 21, 2011
************************************************
```

The first input is an integer that defines the type of trajectory optimization to simulate.

```
type of optimization (1 = maximum payload, 2 = minimum transfer time)
1
```

The next input is an integer that tells the software what type of trajectory to model.

```
trajectory type (1 = flyby, 2 = rendezvous)
2
```

The next input specifies what type of propulsion system to use during the simulation. Option 1 is a constant thrust system and option 2 is a solar-electric propulsion (SEP) system.

```
propulsion type (1 = chemical, 2 = SEP)
1
```

The next input is the initial launch energy. This number is also called C3 which is twice the specific (per unit mass) orbital energy.

```
launch energy ([km/sec]**2)
4.625d0
```

The next input is the initial spacecraft mass, in kilograms.

```
initial spacecraft mass (kilograms)
1171.1
```

The next two inputs define the thrust and specific impulse for a constant thrust, chemical propulsion system (propulsion type option 1).

```
thrust magnitude (newtons)
0.16831
```

```
specific impulse (seconds)
3070.0
```

The next series of inputs define the characteristics of a solar electric propulsion system (propulsion option 2). Please see the technical discussion section later in this document for additional information about these data items and coefficients.

```
solar array power at 1 AU (kw)
5.0

solar array minimum and maximum power (kw)
0.649, 2.6

solar array power coefficients
1.1063, 0.1495, -0.299, -0.0432, 0.0

SEP thrust magnitude coefficients
-1.9137, 36.242, 0.0, 0.0, 0.0

SEP propellant flow rate coefficients
0.47556, 0.90209, 0.0, 0.0, 0.0
```

The lower and upper bounds for the throttle setting are defined by the next two user inputs.

```
lower bound for throttle setting (0 <= bound <= 1)
0.0

upper bound for throttle setting (0 <= bound <= 1)
1.0
```

The software allows the user to specify an initial guess for the launch and arrival calendar dates and lower and upper bounds on the actual dates found during the optimization process. For any guess for launch time t_L and user-defined launch time lower and upper bounds Δt_l and Δt_u, the launch time t is constrained as follows

$$t_L - \Delta t_l \le t \le t_L + \Delta t_u$$

Likewise, for any guess for arrival time t_A and user-defined arrival time bounds Δt_l and Δt_u, the arrival time t is constrained as

$$t_A - \Delta t_l \le t \le t_A + \Delta t_u$$

For fixed launch and/or arrival times, the lower and upper bounds are set to `0`.

The next three inputs are the user's initial guess for the launch calendar date. Be sure to include all four digits of the calendar year.

```
*********************
* LAUNCH CONDITIONS *
*********************

launch calendar date initial guess (month, day, year)
7, 10, 2005
```

The next two inputs are the lower and upper bounds for the launch calendar date search interval. These values should be input in days.

```
launch date search boundary (days)
-60, +60
```

The next program input is an integer that specifies the launch planet.

```
*****************
* launch planet *
*****************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
----------
3
```

The next set of inputs defines the user's initial guess for the arrival calendar date, the arrival date search interval and the arrival celestial body.

```
**********************
* ARRIVAL CONDITIONS *
**********************

arrival calendar date initial guess (month, day, year)
3, 4, 2006

arrival date search boundary (days)
-120, +240

*************************
* arrival celestial body *
*************************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
 0 = asteroid/comet
-------------------
4
```

The next series of inputs include the name and classical orbital elements of a comet or asteroid (arrival celestial body = 0). Please note that the angular orbital elements must be specified with respect to a heliocentric, Earth mean ecliptic and equinox of J2000 coordinate system.

```
*********************************
* asteroid/comet orbital elements *
* (heliocentric, ecliptic J2000)  *
*********************************

asteroid/comet name
Tempel 1

calendar date of perihelion passage (month, day, year)
7, 5.3153, 2005

perihelion distance (au)
1.506167
```

```
orbital eccentricity (nd)
0.517491

orbital inclination (degrees)
10.5301

argument of perihelion (degrees)
178.8390

longitude of the ascending node (degrees)
68.9734
```

This next input defines the type of initial guess to use. Please see the technical discussion section for information about how the first option is modeled. Option 2 requires either a binary "restart" file created from a previous run using initial guess option 1 or an updated binary file. This feature is described in the next two sections.

```
*******************************
* initial guess/restart option *
*******************************
 1 = numerical integration
 2 = binary data file
---------------------
1
```

If the user elects to use a binary data file (option 2 above) for the initial guess, the following text input specifies the name of the file to use.

```
name of initial guess/restart input data file
e2m1_max_payload.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess options `1`, the software will create a binary restart file. For initial guess option `2`, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*****************************
* binary restart file option *
*****************************

create/update binary restart data file (yes or no)
no
```

This next input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by the *Sparse Optimization Suite*. Options 2 and 3 allow the user to specify either the number of nodes (option 2) or time step size of the data file (option 3).

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = OC-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
--------------------------
1
```

For options 2 or 3, this next input defines either the number of data points (option 2) or the time step size of the data output in the solution file (option 3).

```
number of user-defined nodes or print step size in solution data file
1
```

The name of the solution data file is defined in this next line. A description of the information written to this file is summarized later in this document.

```
name of solution output file
e2m1_max_payload.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next input defines the relative error in the objective function. A value of `1.0d-5` is recommended.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations. A value of `1.0d-7` is recommended.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
```

```
2
```

The level of output from the optimal control algorithm is controlled with the following integer input. Please note that option `4` will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the differential equation algorithm is controlled with the following integer input. Please note that option `5` will create lots of information.

```
****************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of inputs allow the reading and writing of configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `ilt_sos` software. Please consult Appendix J – Typical Sparse Optimization Suite Configuration File for additional information about this type of file.

```
************************************
* optimal control configuration options
************************************

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
e2m1_max_payload_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
e2m1_max_payload_config1.txt
```

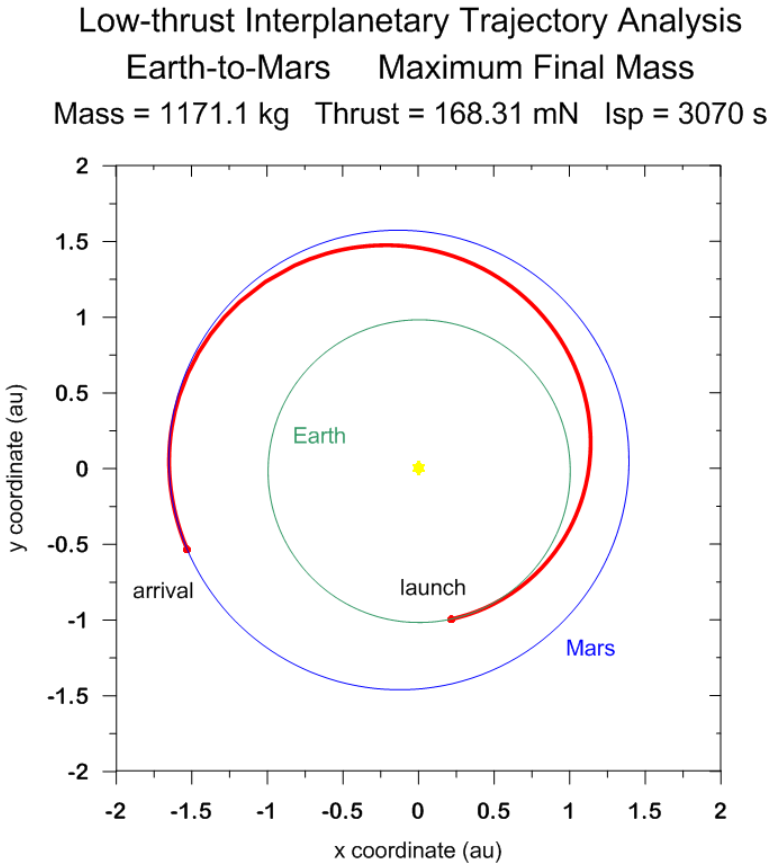
Optimal control solution

The `ilt_sos` software will create two comma-separated-variable (csv) output files. The first file contains the heliocentric, ecliptic state vector of the spacecraft and has the name specified by the user in the main input file. The second file (`planets.csv`) contains the state vectors of both celestial bodies. Additional information about the contents of these two files can be found later in this chapter.

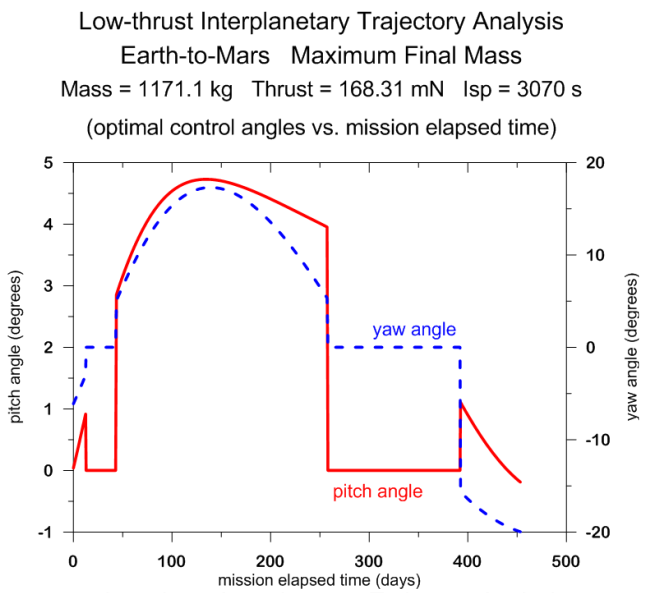
The software will also display a summary of the final solution and a numerical verification of the solution. This information is explained later in this document in the section titled Verification of the optimal control solution.

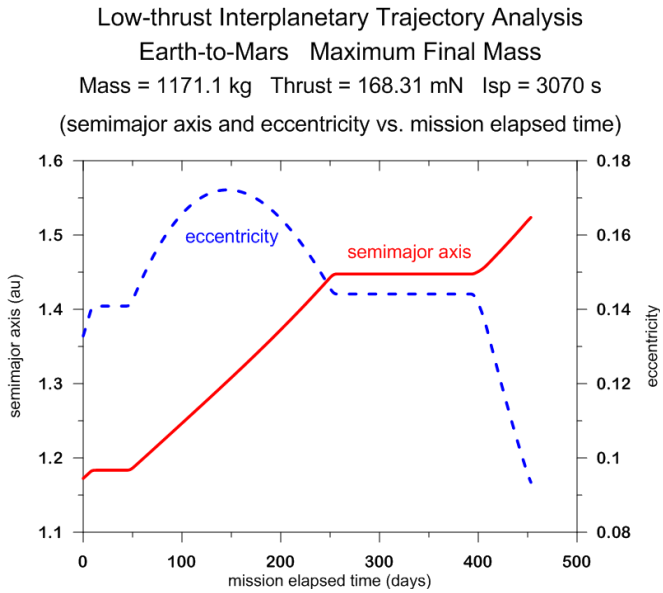
The following are typical transfer trajectory, optimal control and orbital element plots for this example. The first plot is a view of the trajectory and planetary orbits from the north pole of the ecliptic looking down on the ecliptic plane. The second plot illustrates the behavior of the pitch and yaw angles during the orbit transfer. The other plots illustrate the evolution of the heliocentric orbital elements, spacecraft mass, throttle setting and accumulated delta-v during the heliocentric orbital transfer.

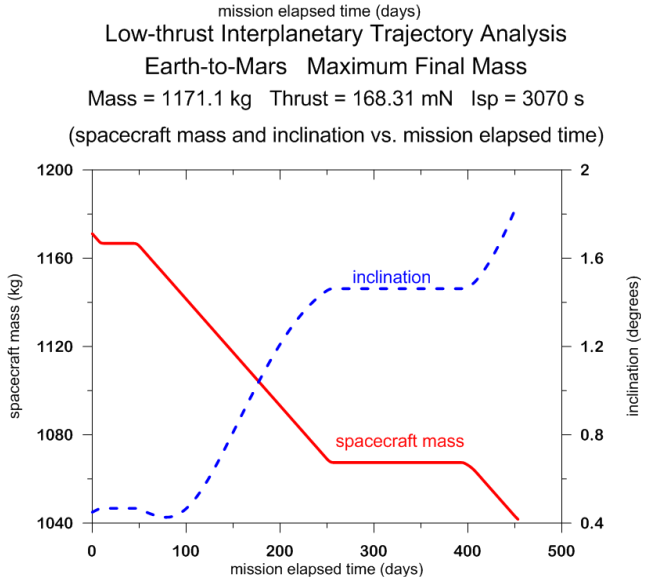
It is interesting to note from these plots that although the simulation is initialized as a single phase, the *Sparse Optimization Suite* determines a multiple maneuver solution. Furthermore, it does this without a priori knowledge about the number of propulsive maneuvers or switching conditions.

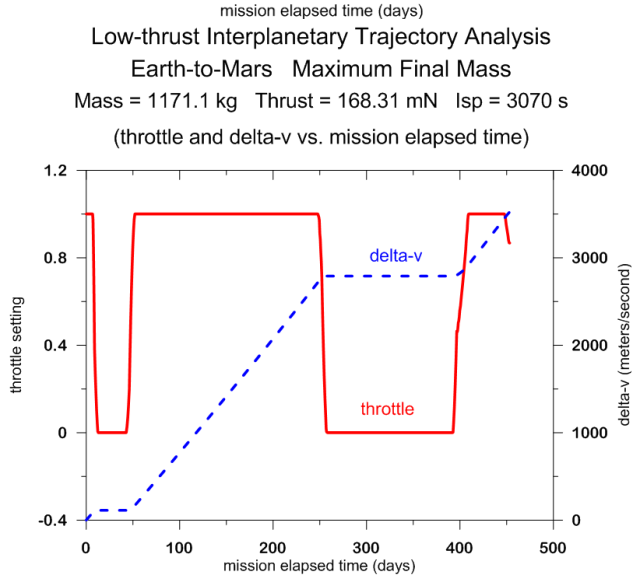


The next three plots are plots that illustrate the behavior of the pitch and yaw angles, the semimajor axis, eccentricity, orbital inclination and throttle setting during the low-thrust orbital transfer.

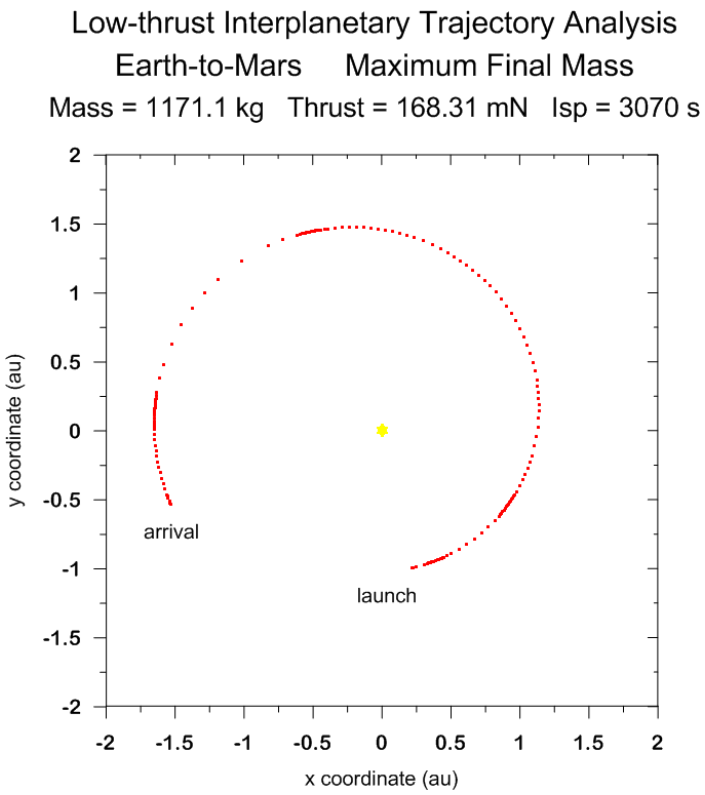








This plot illustrates how the mesh refinement feature of the *Sparse Optimization Suite* has distributed the trajectory nodes of the optimal solution.



Verification of the optimal control solution

The optimal solution determined by the *Sparse Optimization Suite* can be verified by numerically integrating the spacecraft's heliocentric equations of motion using the optimal control computed initial conditions and the optimal control determined by the software. This computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to explicitly integrate the orbital equations of motion.

The following is a typical display of the final solution and errors computed using the explicit numerical integration method. The errors are the differences between the spacecraft's final cartesian state vector and the arrival celestial body's ephemeris. The initial time is `tzero` and the final time is `tfinal`, both measured in days relative to the user-specified launch date initial guess.

```
program ilt_sos

input file ==> e2m1_max_payload.in

rendezvous trajectory

maximum payload

LAUNCH CONDITIONS
-----------------

calendar date       07/04/2005

TDB time            11:12:04.672

TDB julian date     2453555.96672074

launch hyperbola characteristics
(Earth mean equator and equinox of J2000)
----------------------------------------
```

right ascension 14.0052092520307 degrees

declination -1.09827800615456 degrees

launch energy 4.62500000000000 (km/sec)**2

heliocentric orbital elements of the departure planet at launch
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1000769973D+01	0.1595985178D-01	0.1346924405D-02	0.3369525119D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.1264776185D+03	0.1790720860D+03	0.1560245979D+03	0.3656788363D+03

rx (km)	ry (km)	rz (km)	rmag (km)
0.3292667100D+08	-.1484954302D+09	0.1452948353D+04	0.1521021317D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.2860869963D+02	0.6335673965D+01	-.6293294032D-03	0.2930185078D+02

heliocentric orbital elements of the spacecraft at launch
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1172507758D+01	0.1328504477D+00	0.4472981841D+00	0.1797865386D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.1025723223D+03	0.1433534199D+00	0.1799298920D+03	0.4637372842D+03

rx (km)	ry (km)	rz (km)	rmag (km)
0.3292667100D+08	-.1484954302D+09	0.1452948364D+04	0.1521021317D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.3069496859D+02	0.6796705358D+01	-.2454392343D+00	0.3143941063D+02

ARRIVAL CONDITIONS
---

calendar date 09/30/2006

TDB time 13:12:05.013

TDB julian date 2454009.05005802

flight time 453.083337281990 days

spacecraft mass 1041.74010208421 kilograms

delta-v 3523.98568030763 meters/second

heliocentric conditions of the destination celestial body at arrival
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1523685751D+01	0.9343043699D-01	0.1849305087D+01	0.2865593844D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.4953837629D+02	0.2229171805D+03	0.1494765650D+03	0.6869759546D+03

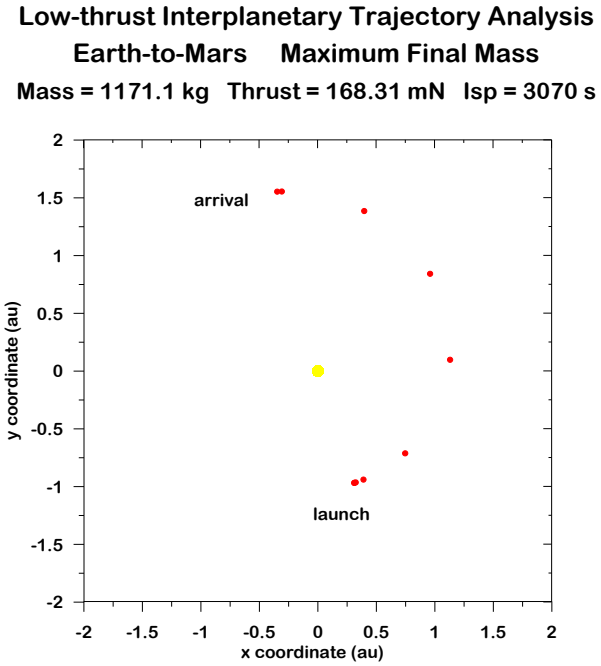
```
     rx (km)              ry (km)              rz (km)           rmag (km)
 -.2292623909D+09    -.7906671107D+08    0.3975347044D+07    0.2425460618D+09

     vx (kps)             vy (kps)             vz (kps)          vmag (kps)
 0.8805666405D+01    -.2083594843D+02    -.6528878327D+00    0.2262968781D+02

heliocentric conditions of the spacecraft at arrival
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.1523685751D+01    0.9343043699D-01    0.1849305087D+01    0.2865593844D+03

    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.4953837629D+02    0.2229171805D+03    0.1494765650D+03    0.6869759546D+03


     rx (km)              ry (km)              rz (km)           rmag (km)
 -.2292623909D+09    -.7906671107D+08    0.3975347044D+07    0.2425460618D+09

     vx (kps)             vy (kps)             vz (kps)          vmag (kps)
 0.8805666405D+01    -.2083594843D+02    -.6528878327D+00    0.2262968781D+02

INTEGRATED SOLUTION WITH OPTIMAL CONTROL
========================================

tzero               -5.53327926122688      days

tfinal               447.550058020763      days

final heliocentric position vector and magnitude errors

delta rx             582.767748981714      kilometers
delta ry            -783.020385280252      kilometers
delta rz            -19.9336046096869      kilometers

delta rmag           976.287110234681      kilometers


final heliocentric velocity vector and magnitude errors

delta vx           8.919249560612741E-005 kilometers/second
delta vy           1.474770563802963E-005 kilometers/second
delta vz          -1.316307802645689E-006 kilometers/second

delta vmag         9.041310060088822E-005 kilometers/second

delta-v             3527.31138777129      meters/second
```

Creating an initial guess

An initial guess for the algorithm is created by numerically integrating the modified equinoctial equations of motion for a user-defined orbital transfer time. During the interplanetary phase to an *outer planet*, the algorithm assumes *tangential thrusting* such that the unit thrust vector in the modified equinoctial frame at all times is simply $\mathbf{u}_T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$. Please note that this approach creates a *coplanar* initial guess. However, since the orbital inclinations of most solar system objects of interest are relatively small, the software has little trouble finding a feasible trajectory and eventually an optimized solution. This initial guess algorithm uses the launch date, arrival date, propulsive characteristics and spacecraft mass provided by the user. For transfer to an *inner planet*, the unit thrust vector is $\mathbf{u}_T = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^T$. A throttle setting of one is used during the initial guess computations.

The dynamic variables and control variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine. These program options create an initial guess from the numerical integration of the equations programmed in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 2` program option tells the algorithm to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

The following plot illustrates the location of typical trajectory grid points using this technique. This initial guess consists of ten grid points. These grid points are located at the integration steps determined by the Dormand-Prince algorithm.



To model an impulsive delta-v at launch, the initial guess algorithm assumes that this maneuver is initially applied along the direction of the velocity vector of the departure planet. For outer planet missions, this direction is along the direction of orbital motion. For inner planet missions, this vector is aligned opposite to the direction of orbital motion.

The unit thrust vector of this maneuver in the heliocentric inertial coordinate system is $\mathbf{u}_T = \mathbf{r}_p / |\mathbf{r}_p|$ where $\mathbf{r}_p$ is the heliocentric inertial position vector of the departure planet at launch. Since the modified equinoctial form of the unit thrust vector is the optimal control in this example, the *SOS* software will change this unit pointing vector while searching for a solution.

An initial guess for the transfer time can be created by performing a one-dimensional minimization while numerically integrating the equations of motion with tangential thrusting. This process determines future close approach conditions between the spacecraft in its *coplanar* transfer orbit and the destination celestial body. This computational process can be performed by a utility computer program that uses a minimization algorithm with the following objective function

$$f(t) = \Delta r(t) = \left| \mathbf{r}_p - \mathbf{r}_{sc} \right|$$

where $\mathbf{r}_p$ is the heliocentric position vector of the arrival body and $\mathbf{r}_{sc}$ is the heliocentric position vector of the spacecraft at any simulation time *t*. If the separation distance Δr is "close" enough (say 0.01 to perhaps 0.1 AU), the trajectory time provides a good initial guess. The close approach algorithm also uses the launch date, propulsive characteristics and spacecraft mass provided by the user. For consistency, this software also uses the same planetary ephemeris as the `ilt_sos` computer program.

Problem setup

This section provides additional details about the software implementation. For good scaling, the time unit used in all internal calculations is days, position is expressed in astronomical units and the velocity unit is astronomical units per day.

Launch and arrival time bounds

The software allows the user to specify an initial guess for the launch and arrival calendar dates and lower and upper bounds on the actual dates found during the optimization process. For any guess for launch time t_L and user-defined launch time lower and upper bounds Δt_l and Δt_u, the launch time t is constrained as follows

$$t_L - \Delta t_l \le t \le t_L + \Delta t_u$$

Likewise, for any guess for arrival time t_A and user-defined arrival time bounds Δt_l and Δt_u, the arrival time t is constrained as follows:

$$t_A - \Delta t_l \le t \le t_A + \Delta t_u$$

For fixed launch and/or arrival times, the lower and upper bounds are set to `0`.

Performance index

The objective function or performance index *J* for this simulation is either the final mass of the spacecraft m_f or the total transfer time t_f. This is simply $J = m_f$ or $J = t_f$.

The value of the `maxmin` indicator tells the *SOS* software whether the user is minimizing or maximizing the performance index. The spacecraft mass at the initial time is fixed to the initial value. However, the `ilt_sos` software can be easily modified to make the initial mass the performance index.

Path constraint – unit thrust vector scalar magnitude

At any point during the interplanetary transfer trajectory, the scalar magnitude of the components of the unit thrust vector is constrained as follows

$$\left| \mathbf{u}_T \right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

This formulation avoids problems that may occur when using thrust steering angles as control variables.

Point functions – position and velocity vector "matching" at launch

For any launch time t_L the optimal solution must satisfy the following state vector boundary conditions (equality constraints) at launch

$$\mathbf{r}_{s/c}(t_L) - \mathbf{r}_p(t_L) = 0$$

$$\mathbf{v}_{s/c}(t_L) - \left\{\mathbf{v}_p(t_L) + \Delta\mathbf{v}(t_L)\right\} = 0$$

where $\mathbf{r}_{s/c}$ and $\mathbf{v}_{s/c}$ are the heliocentric, inertial position and velocity vectors of the spacecraft at the launch time t_L, $\mathbf{r}_p$ and $\mathbf{v}_p$ are the heliocentric, inertial position and velocity vectors of the departure planet at the launch time, and $\Delta\mathbf{v}$ is the user-specified available *impulsive* delta-v vector at launch. This delta-v might be provided by the launch vehicle or perhaps a high-thrust upper stage. The delta-v scalar magnitude is equal to the square root of the launch specific orbital energy, also called C3L.

In this constraint formulation, the delta-v contribution is modeled as $\Delta\mathbf{v} = |\Delta\mathbf{v}|\hat{\mathbf{u}}_T$ where $\hat{\mathbf{u}}_T$ is the unit thrust vector in the heliocentric, inertial coordinate system. Since the optimal control for this problem is the unit thrust vector in the modified equinoctial system, $\hat{\mathbf{u}}_T$ is computed from

$$\hat{\mathbf{u}}_T = \left[\hat{\mathbf{i}}_r \quad \hat{\mathbf{i}}_t \quad \hat{\mathbf{i}}_n\right]\hat{\mathbf{u}}_{mee}$$

where $\hat{\mathbf{u}}_{mee}$ is the unit thrust vector in the modified equinoctial coordinate system and

$$\hat{\mathbf{i}}_r = \frac{\mathbf{r}}{|\mathbf{r}|} \qquad \hat{\mathbf{i}}_n = \frac{\mathbf{r}\times\mathbf{v}}{|\mathbf{r}\times\mathbf{v}|} \qquad \hat{\mathbf{i}}_t = \hat{\mathbf{i}}_n \times \hat{\mathbf{i}}_r = \frac{(\mathbf{r}\times\mathbf{v})\times\mathbf{r}}{|\mathbf{r}\times\mathbf{v}||\mathbf{r}|}$$

Point functions – position and velocity vector "matching" at arrival

For any arrival time t_A the optimal solution must satisfy the following state vector boundary conditions (BC) at arrival

$$\mathbf{r}_{s/c}(t_A) - \mathbf{r}_p(t_A) = 0$$

$$\mathbf{v}_{s/c}(t_A) - \mathbf{v}_p(t_A) = 0$$

where $\mathbf{r}_{s/c}$ and $\mathbf{v}_{s/c}$ are the heliocentric, inertial position and velocity vectors of the spacecraft at the arrival time t_A, and $\mathbf{r}_p$ and $\mathbf{v}_p$ are the heliocentric, inertial position and velocity vectors of the destination planet at the arrival time. This system of launch and arrival state vector equality constraints ensures a rendezvous mission. For a flyby mission, the velocity vector point functions at arrival are <u>not</u> enforced by the software.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables during the interplanetary orbital transfer.

$$0.05m_{sc_i} \le m_{sc} \le 1.05m_{sc_i} \qquad 0.5r_p \le p \le 2r_a$$

$$-1 \le f \le +1 \qquad -1 \le g \le +1 \qquad -1 \le h \le +1 \qquad -1 \le k \le +1$$

where r_p is the heliocentric periapsis radius of the initial orbit, r_a is the apoapsis radius of the final orbit, and m_{sc_i} is the initial spacecraft mass provided by the user. These two distances, in astronomical units, are determined from the orbital elements of the departure and arrival bodies at the user's initial guess for the corresponding calendar dates. Since we may allow the *Sparse Optimization Suite* to change the actual launch time during the optimization, the true longitude is unconstrained.

The components of the unit thrust vector are bounded as follows

$$-1.1 \le u_r \le +1.1 \qquad -1.1 \le u_t \le +1.1 \qquad -1.1 \le u_n \le +1.1$$

Finally, the throttle setting is bounded to user-defined lower and upper bounds according to $\eta_L \le \eta \le \eta_U$ where these bounds are typically between 0 and 1.

The natural bounds for this trajectory formulation are ideal for numerical optimization.

Technical discussion

The motion of the spacecraft is modeled in the modified equinoctial orbital element system. Please consult *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for the equations of motion and coordinate conversions.

Planetary ephemeris

The software models the planetary coordinates using the DE 421 model from JPL. This planetary ephemeris provides position and velocity vectors relative to the Earth mean equator and equinox of J2000 (EME2000). The `ilt_sos` software converts the DE 421 state vectors to the Earth mean ecliptic system using the following transformation

$$\mathbf{S}_{ec} = \begin{bmatrix} 1 & -0.000000479966 & 0 \\ 0.000000440360 & 0.917482137087 & 0.397776982902 \\ -0.000000190919 & -0.397776982902 & 0.917482137087 \end{bmatrix}^T \mathbf{S}_{eq}$$

where $\mathbf{S}_{eq}$ is the state vector in the Earth mean equatorial frame, and $\mathbf{S}_{ec}$ is the state vector in the Earth mean ecliptic frame.

Important Note

The binary ephemeris file provided with this computer program was created for use on Windows compatible computers. For other platforms, you will need to create or obtain binary files specific to that system. Information and computer programs for creating these files can be found at the JPL solar system FTP site located at ftp://ssd.jpl.nasa.gov/pub/eph/export/. This site provides ASCII data files and Fortran computer programs for creating a binary file. A program for testing the user's ephemeris is also provided along with documentation.

Asteroid and comet ephemeris

Please consult *Appendix F – Computing an Asteroid or Comet Ephemeris* for the numerical method used in this software to compute small body ephemerides.

Solar electric propulsion (SEP)

For the SEP option, the power available, in kilowatts, for the engine is computed from

$$P = \frac{P_0}{r^2}\left(\frac{a_1 + \frac{a_2}{r} + \frac{a_3}{r^2}}{1 + a_4 r + a_5 r^2}\right)$$

where P_0 is the power available at one astronomical unit (AU), *r* is the heliocentric distance of the spacecraft, in astronomical units, and the a_i *'s* are coefficients provided by the user.

The propulsive thrust *T* and mass flow rate $\dot{m}$ are computed from the following polynomials which are functions of the available power *P*.

$$T = c_1 + c_2 P + c_3 P^2 + c_4 P^3 + c_5 P^4$$

$$\dot{m} = d_1 + d_2 P + d_3 P^2 + d_4 P^3 + d_5 P^4$$

The coefficients for each polynomial are provided by the user. In these equations, the fundamental unit is milli-Newtons for thrust and milligrams for spacecraft mass. Finally, the software will also constrain the input power to the electric engine to the user-defined lower and upper bounds according to

$$P_L \le P \le P_U$$

Launch characteristics

For interplanetary missions, the orientation of the departure hyperbola is specified in terms of the right ascension and declination of the outgoing asymptote. These coordinates can be calculated using the components of the initial spacecraft and departure planet heliocentric velocity vectors determined by the *Sparse Optimization Suite*.

In this computer program, the heliocentric planetary coordinates and the transfer orbit velocity vectors are computed in the Earth mean ecliptic and equinox of J2000 coordinate system. To determine the orientation of the departure hyperbola in the EME2000 system, the initial heliocentric velocity vector must be transformed to the equatorial frame.

The required matrix-vector transformation is given by

$$\mathbf{V}_{\infty_{EQ}} = \begin{bmatrix} 1 & -0.000000479966 & 0 \\ 0.000000440360 & 0.917482137087 & 0.397776982902 \\ -0.000000190919 & -0.397776982902 & 0.917482137087 \end{bmatrix} \mathbf{V}_{\infty_{EC}}$$

where $\mathbf{V}_{\infty_{EC}}$ is the v-infinity velocity vector in the ecliptic frame, and $\mathbf{V}_{\infty_{EQ}}$ is the v-infinity velocity vector in the equatorial frame. The ecliptic v-infinity velocity vector can be computed using

$$\mathbf{V}_{\infty_{EC}} = \mathbf{V}_{sc} - \mathbf{V}_{dp}$$

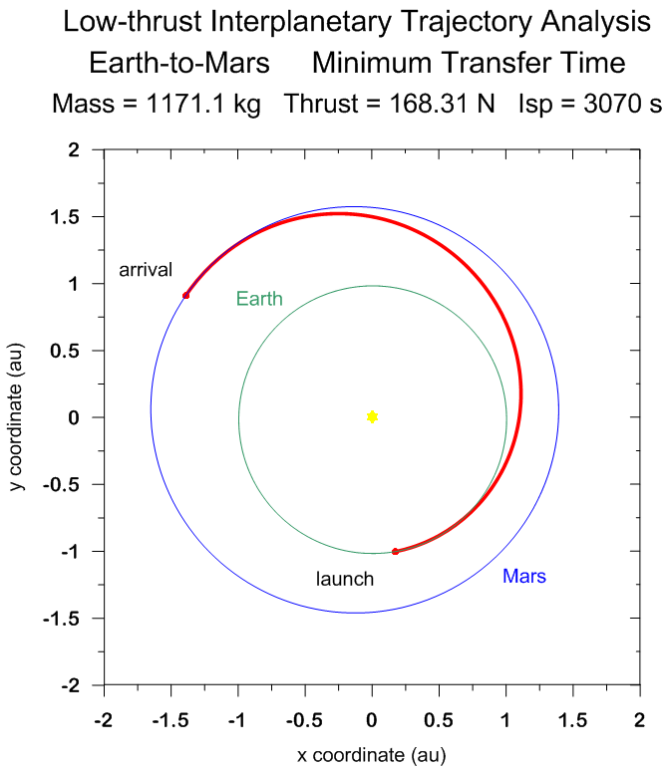
where $\mathbf{V}_{sc}$ is the heliocentric, ecliptic velocity vector of the spacecraft and $\mathbf{V}_{dp}$ is the velocity vector of the departure planet, also in the ecliptic frame.

The right ascension of the asymptote is determined from $\alpha = \tan^{-1}\left(V_y, V_z\right)$ and the geocentric declination of the outgoing asymptote is given by $\delta = 90^0 - \cos^{-1}\left(\hat{V}_z\right)$ where $\hat{V}_z$ is the z-component of the unit v-infinity velocity vector in the equatorial frame of reference. The right ascension is computed using a four-quadrant inverse tangent function.

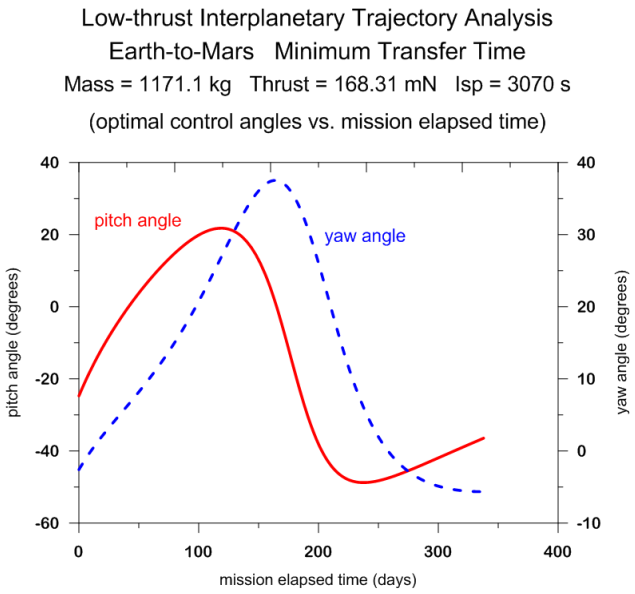
Please consult *Appendix H – Impulsive Interplanetary Injection from a Circular Earth Orbit* for additional information and equations related to this aspect of interplanetary mission analysis.

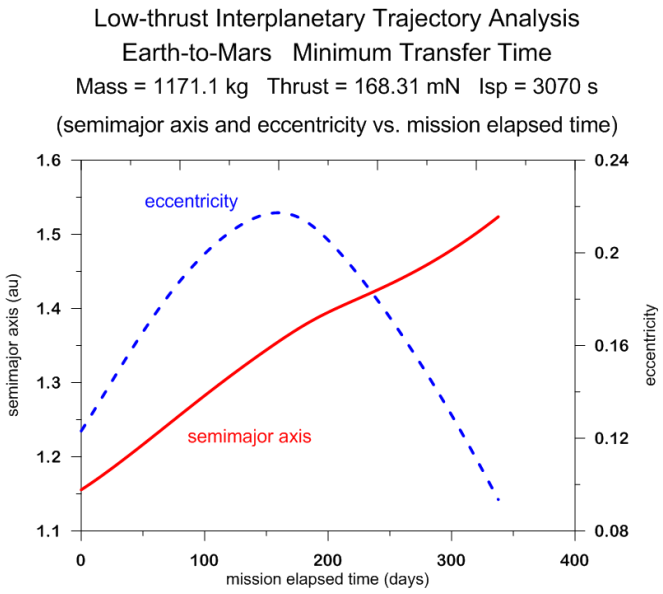
Example minimum transfer time trajectory analysis

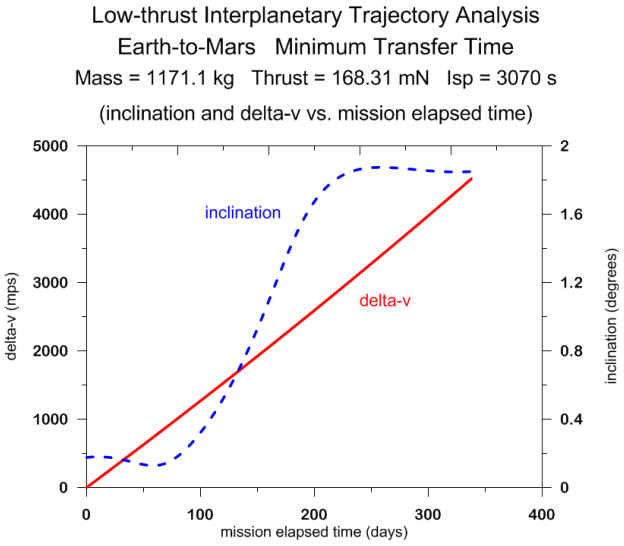
This section contains graphics and a simulation summary for a typical Earth-to-Mars *minimum transfer time* interplanetary rendezvous mission using chemical propulsion.



The following plots illustrate the behavior of the maneuver pitch and yaw angles, the heliocentric semimajor axis, eccentricity, orbital inclination and accumulated delta-v during the interplanetary transfer.







Here is the numerical output for this example using trapezoidal collocation.

```
program ilt_sos

input file ==> e2m1_min_time.in

rendezvous trajectory

minimum transfer time

LAUNCH CONDITIONS
-----------------

calendar date       07/01/2005

TDB time            11:26:09.314

TDB julian date     2453552.97649669

launch hyperbola characteristics
(Earth mean equator and equinox of J2000)
----------------------------------------

right ascension     33.0041733115229      degrees
```

```
declination          10.6195596448380      degrees

launch energy        4.62500000000000      (km/sec)**2

heliocentric orbital elements of the departure planet at launch
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.1000432031D+01    0.1631213315D-01    0.2205231122D-02    0.3477645136D+03

    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.1163335586D+03    0.1755526005D+03    0.1633171141D+03    0.3654936272D+03

     rx (km)             ry (km)             rz (km)            rmag (km)
 0.2549751741D+08    -.1499437805D+09    0.1680522702D+04    0.1520962219D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 0.2888957025D+02    0.4874377540D+01    -.1079752277D-02    0.2929789799D+02

heliocentric orbital elements of the spacecraft at launch
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.1155321712D+01    0.1230681108D+00    0.1749818469D+00    0.1943199109D+03
    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.9985796247D+02    0.3454727984D+03    0.1797927093D+03    0.4535788853D+03


     rx (km)             ry (km)             rz (km)            rmag (km)
 0.2549751741D+08    -.1499437805D+09    0.1680522702D+04    0.1520962219D+09


     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 0.3066222342D+02    0.6088377602D+01    -.9544380167D-01    0.3126098841D+02
```

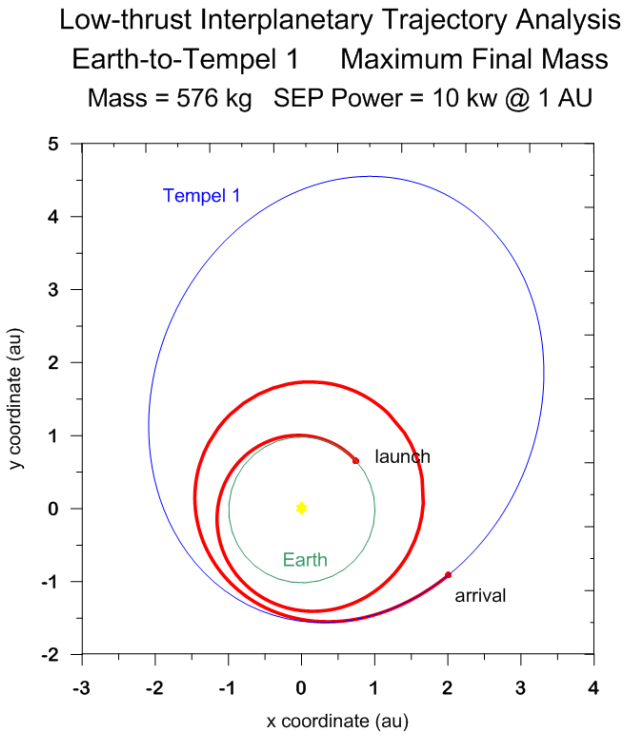
ARRIVAL CONDITIONS

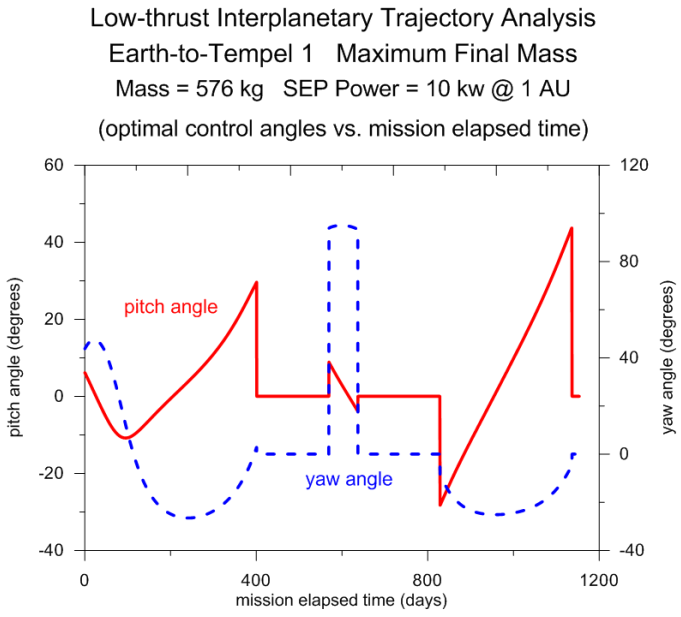
```
------------------

calendar date        06/04/2006

TDB time             07:31:46.918

TDB julian date      2453890.81373747


flight time          337.837240782112      days

spacecraft mass      1007.91804464375      kilograms

delta-v              4517.66684906135      meters/second

heliocentric conditions of the destination celestial body at arrival
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.1523593257D+01    0.9348701651D-01    0.1849328659D+01    0.2865824155D+03

    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.4953748812D+02    0.1705051279D+03    0.9708754335D+02    0.6869134021D+03

     rx (km)             ry (km)             rz (km)            rmag (km)
 -.2077412613D+09    0.1368309357D+09    0.7970392623D+07    0.2488828314D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 -.1241641781D+02    -.1816677609D+02    -.7563626703D-01    0.2200465645D+02

heliocentric conditions of the spacecraft at arrival
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.1523593257D+01    0.9348701651D-01    0.1849328659D+01    0.2865824155D+03
```

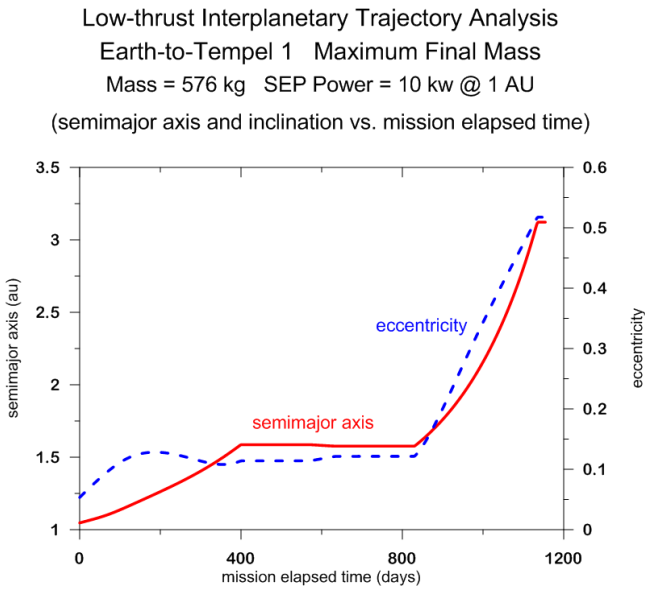
```
    raan (deg)       true anomaly (deg)      arglat (deg)         period (days)
 0.4953748812D+02    0.1705051279D+03    0.9708754335D+02    0.6869134021D+03

     rx (km)             ry (km)             rz (km)             rmag (km)
 -.2077412613D+09    0.1368309357D+09    0.7970392623D+07    0.2488828314D+09

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 -.1241641781D+02    -.1816677609D+02    -.7563626703D-01    0.2200465645D+02

 INTEGRATED SOLUTION WITH OPTIMAL CONTROL
 ========================================

 tzero               -8.52350330775228      days

 tfinal               329.313737474360      days

 final heliocentric position vector and magnitude errors

 delta rx            0.982853084802628      kilometers
 delta ry            -1.99446716904640      kilometers
 delta rz           -0.300119932740927      kilometers

 delta rmag           2.24365136523891      kilometers

final heliocentric velocity vector and magnitude errors

 delta vx            1.845999761940220E-007 kilometers/second
 delta vy           -2.231874027813774E-007 kilometers/second
 delta vz           -2.566047183072406E-008 kilometers/second

 delta vmag          2.907717795552135E-007 kilometers/second

 delta-v              4517.66684916448      meters/second
```

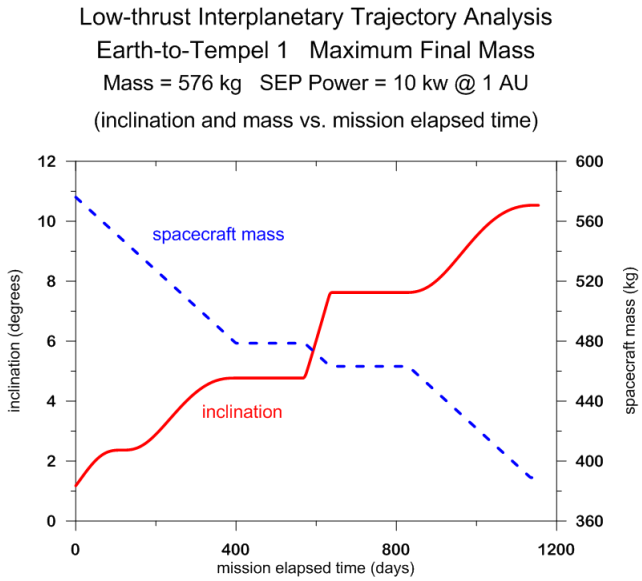
Example comet rendezvous trajectory analysis

This section contains graphics and a simulation summary for a maximum final mass, interplanetary rendezvous mission from Earth to the comet Tempel 1 using solar electric propulsion.

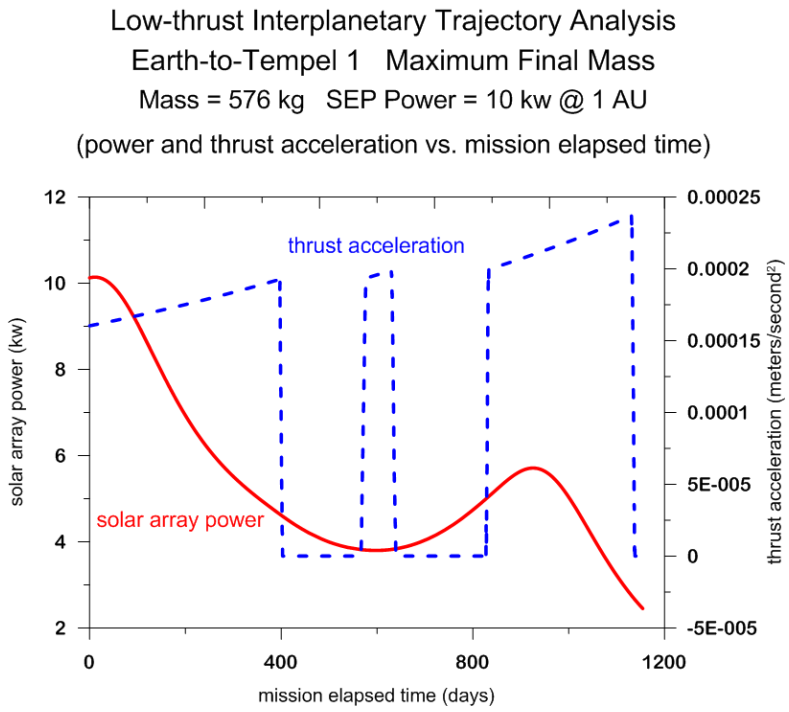








The final plot illustrates the behavior of the available solar array power and the thrust acceleration during the interplanetary transfer.



Here's the program numerical output for this example using trapezoidal collocation.

```
program ilt_sos

input file ==> tempel1_max_payload.in

rendezvous trajectory

maximum payload

LAUNCH CONDITIONS
-----------------
```

calendar date 11/04/2002

TDB time 01:07:29.851

TDB julian date 2452582.54687327

launch hyperbola characteristics
(Earth mean equator and equinox of J2000)
---

right ascension 145.002380710242 degrees

declination 61.9245330434051 degrees

launch energy 0.800000000000000 (km/sec)**2

heliocentric orbital elements of the departure planet at launch
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1000926361D+01	0.1706849123D-01	0.1505127320D-02	0.4459557462D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.5564889164D+02	0.3011632584D+03	0.3457588330D+03	0.3657645552D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.1112899240D+09	0.9814193557D+08	-.9589024794D+03	0.1483822318D+09
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.2019384909D+02	0.2224169878D+02	0.7676480465D-03	0.3004138323D+02

heliocentric orbital elements of the spacecraft at launch
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.1046435806D+01	0.5329831146D-01	0.1174576604D+01	0.1259590030D+02
raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.4142578374D+02	0.3473860368D+03	0.3599819371D+03	0.3909914983D+03
rx (km)	ry (km)	rz (km)	rmag (km)
0.1112899240D+09	0.9814193557D+08	-.9589024815D+03	0.1483822318D+09
vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.2053867962D+02	0.2277712479D+02	0.6287890598D+00	0.3067621473D+02

ARRIVAL CONDITIONS
---

calendar date 12/31/2005

TDB time 09:27:25.212

TDB julian date 2453735.89404180

flight time 1153.34716852868 days

spacecraft mass 388.957318524463 kilograms

delta-v 12848.8672295688 meters/second

heliocentric conditions of the destination celestial body at arrival
(Earth mean ecliptic and equinox of J2000)
---

sma (au)	eccentricity	inclination (deg)	argper (deg)
0.3121939735D+01	0.5175674802D+00	0.1052963297D+02	0.1788381138D+03
raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
0.6894110071D+02	0.8761888543D+02	0.2664569992D+03	0.2014815113D+04

```
     rx (km)             ry (km)             rz (km)            rmag (km)
 0.2990939734D+09    -.1373302895D+09    -.6105302696D+08    0.3347301681D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 0.1746174759D+02    0.1412553576D+02    -.2085471820D+01    0.2255643106D+02

heliocentric conditions of the spacecraft at arrival
(Earth mean ecliptic and equinox of J2000)
-----------------------------------------

     sma (au)          eccentricity       inclination (deg)     argper (deg)
 0.3121939735D+01    0.5175674802D+00    0.1052963297D+02    0.1788381138D+03

    raan (deg)       true anomaly (deg)     arglat (deg)        period (days)
 0.6894110071D+02    0.8761888543D+02    0.2664569992D+03    0.2014815113D+04


     rx (km)             ry (km)             rz (km)            rmag (km)
 0.2990939734D+09    -.1373302895D+09    -.6105302696D+08    0.3347301681D+09

     vx (kps)            vy (kps)            vz (kps)           vmag (kps)
 0.1746174759D+02    0.1412553576D+02    -.2085471820D+01    0.2255643106D+02

INTEGRATED SOLUTION WITH OPTIMAL CONTROL
========================================

tzero              -26.9531267251387      days
tfinal              1126.39404180354      days


final heliocentric position vector and magnitude errors

delta rx            4.34277135133743      kilometers
delta ry           -27.3916816115379      kilometers
delta rz            8.84477781504393      kilometers

delta rmag          29.1100322760961      kilometers


final heliocentric velocity vector and magnitude errors

delta vx           2.650010788585178E-006 kilometers/second
delta vy          -1.123592202390000E-006 kilometers/second
delta vz          -3.619323889481052E-007 kilometers/second

delta vmag         2.901036309848453E-006 kilometers/second

delta-v             12879.4667192551      meters/second
```

Here's the input data file for this example.

```
*****************************************
** interplanetary trajectory optimization
** patched-conic, n-body heliocentric motion
** low-thrust maneuvers
** Earth-to-Tempel 1 - March 21, 2011
*****************************************

type of optimization (1 = maximum payload, 2 = minimum transfer time)
1

trajectory type (1 = flyby, 2 = rendezvous)
2

propulsion type (1 = chemical, 2 = SEP)
2

launch energy ([km/sec]**2)
0.8

initial spacecraft mass (kilograms)
576.0
```

```
thrust magnitude (newtons)
0.1

specific impulse (seconds)
3370.0

solar array power at 1 AU (kw)
10.0

solar array minimum and maximum power (kw)
0.649, 2.6

solar array power coefficients
1.1063, 0.1495, -0.299, -0.0432, 0.0

SEP thrust magnitude coefficients
-1.9137, 36.242, 0.0, 0.0, 0.0

SEP propellant flow rate coefficients
0.47556, 0.90209, 0.0, 0.0, 0.0

lower bound for throttle setting (0 <= bound <= 1)
0.0

upper bound for throttle setting (0 <= bound <= 1)
1.0

*********************
* LAUNCH CONDITIONS *
*********************

launch calendar date initial guess (month, day, year)
12,1,2002

launch date search boundary (days)
-30, +30

*****************
* launch planet *
*****************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
----------
3

**********************
* ARRIVAL CONDITIONS *
**********************

arrival calendar date initial guess (month, day, year)
12,3,2005

arrival date search boundary (days)
-60, +60

*************************
* arrival celestial body *
*************************
 1 = Mercury
 2 = Venus
 3 = Earth
 4 = Mars
 5 = Jupiter
 6 = Saturn
 7 = Uranus
 8 = Neptune
 9 = Pluto
 0 = asteroid/comet
```

```
-------------------
0

**********************************
* asteroid/comet orbital elements *
* (heliocentric, ecliptic J2000)  *
**********************************

asteroid/comet name
Tempel 1

calendar date of perihelion passage (month, day, year)
7,5.31630136,2005

perihelion distance (au)
1.50612525322912

orbital eccentricity (nd)
0.517567480182153

orbital inclination (degrees)
10.5296329695782

argument of perihelion (degrees)
178.83811381255987

longitude of the ascending node (degrees)
68.94110070770958

*******************************
* initial guess/restart option *
*******************************
 1 = numerical integration
 2 = binary data file
---------------------
1

name of initial guess/restart input data file
tempel1_max_payload.rsbin

****************************
* restart data file option *
****************************

create/update binary restart data file (yes or no)
no


*********************************************
* type of comma-delimited solution data file *
*********************************************
 1 = OC-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1

number of user-defined nodes or print step size in solution data file
1

name of solution output file
tempel1_max_payload.csv

*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

```
relative error in the objective function (performance index)
1.0d-5

relative error in the solution of the differential equations
1.0d-7

maximum number of mesh refinement iterations
20

maximum number of function evaluations
50000

maximum number of algorithm iterations
5000

**************************
sparse NLP iteration output
--------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2

**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1

***************************
differential equation output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1

******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0

**************************************
* optimal control configuration options
**************************************

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
tempel1_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
tempel1_config.txt
```

Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `ilt_sos` software. All output is computed and displayed in a heliocentric, Earth mean ecliptic and equinox of J2000 coordinate system.

The simulation summary screen display contains the following information:

calendar date = calendar date of trajectory event

ephemeris time = ephemeris time of trajectory event

julian date = julian date of trajectory event

sma (au) = semimajor axis in astronomical unit

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (days) = orbital period in days

delta-v = scalar magnitude of the low-thrust maneuver in meters/seconds

The accumulated delta-v is computed from a cubic spline integration of the thrust acceleration at all grid points determined by the software.

The comma-separated-variable disk file is created by the `odeprt` subroutine and contains the following information.

time (days) = simulation time since launch in days

semimajor axis (au) = semimajor axis in astronomical units

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

arg of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

pitch = thrust vector pitch angle in degrees

yaw = thrust vector yaw angle in degrees

mass = spacecraft mass in kilograms

thracc = thrust acceleration in meters/second**2

rx (au) = x-component of the spacecraft's heliocentric position vector in astronomical units

ry (au) = y-component of the spacecraft's heliocentric position vector in astronomical units

rz (au) = z-component of the spacecraft's heliocentric position vector in astronomical units

vx (km/sec) = x-component of the spacecraft's heliocentric velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's heliocentric velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's heliocentric velocity vector in kilometers per second

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

pmee = orbital semiparameter

fmee = modified equinoctial orbital element = ecc * cos(argper + raan)

gmee = modified equinoctial orbital element = ecc * sin(argper + raan)

hmee = modified equinoctial orbital element = tan(i/2) * cos(raan)

xkmee = modified equinoctial orbital element = tan(i/2) * sin(raan)

xlmee = true longitude in degrees

deltav (mps) = accumulative delta-v in meters per second

throttle = throttle setting

thrust = propulsive thrust (Newtons)

The `planets.csv` file contains the following information.

time (days) = simulation time since launch in days

rp1-x (au) = x-component of the launch planet heliocentric position vector in astronomical units
rp1-y (au) = y-component of the launch planet heliocentric position vector in astronomical units
rp1-z (au) = z-component of the launch planet heliocentric position vector in astronomical units
rp2-x (au) = x-component of the destination body heliocentric position vector in astronomical units
rp2-y (au) = y-component of the destination body heliocentric position vector in astronomical units
rp2-z (au) = z-component of the destination body heliocentric position vector in astronomical units

An Introduction to the Mathematics and Methods of Astrodynamics, Richard H. Battin, AIAA Education Series, 1987.

"Optimal Interplanetary Orbit Transfers by Direct Transcription", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 42, No. 3, July-September 1994, pp. 247-268.

"Interplanetary Mission Design Handbook, Volume 1, Part 2", JPL Publication 82-43, September 15, 1983.

"Interplanetary Mission Analysis and Design", Stephen Kemble, Praxis Publishing, 2006.

"Fuel-Optimal, Low-Thrust, Three-Dimensional Earth-Mars Trajectories", R. S. Nah, S. R. Vadali, and E. Braden, *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 6, November-December 2001.

"Optimal Low-Thrust Interception of Earth-Crossing Asteroids", Bruce A. Conway, *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 5, September-October 1997.

"Electric Propulsion Mission Analysis", NASA SP-210, 1969.

"Possibilities of Combining High- and Low-Thrust Engines in Flights to Mars", G. G. Fedotov, *Cosmic Research*, Vol. 39, No. 6, 2001, pp. 613-621.

"Rapid Solar Sail Rendezvous Missions to Asteroid 99942 Apophis", Giovanni Mengali and Alessandro A. Quarta, AIAA *Journal of Spacecraft and Rockets*, Vol. 46, No. 1, January-February 2009, pp. 134-140.

"The Planetary and Lunar Ephemeris DE 421", W. M. Folkner, J. G. Williams, D. H. Boggs, JPL IOM 343R-08-003, 31-March-2008.

"IERS Conventions (2003)", IERS Technical Note 32, November 2003.

"Planetary Constants and Models", R. Vaughan, JPL D-12947, December 1995.

CMATO 15 – Interplanetary Trajectory Correction Maneuver Optimization

This *CMATO* application is a Fortran computer program named `itcm_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the classic one impulse interplanetary trajectory correction maneuver (TCM) optimization problem. The software attempts to minimize the scalar magnitude of the TCM delta-v vector for user-defined final orbit conditions at Mars.

The important features of this computer implementation are

- heliocentric, inertial cartesian equations of motion with point-mass planetary perturbations
- elliptical, non-coplanar planetary orbits (JPL DE421 ephemeris model)
- hybrid (implicit/explicit) solution of the spacecraft equations of motion

The final conditions at Mars can be determined using one of the following user-defined options.

- flight path angle, radius and orbital inclination at entry interface (EI)
- user-defined B-plane coordinates
- grazing flyby; user-defined B-plane angle

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods.

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `itcm_sos` software consists of Fortran routines that perform the following tasks.

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method.

The `itcm_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `itcm_sos` computer program is "data-driven" by a simple user-created text file. The following is a typical input or "simulation definition" file used by the software. This example is an Earth-to-Mars trajectory that begins at the Earth's sphere-of-influence (SOI) and ends at hyperbolic encounter with Mars. It can be found as `e2m_tcm.in` in the software distribution.

In the following discussion the actual input file contents are in *courier* font and all explanations are in times font. Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is <u>not</u> case sensitive but must be spelled correctly.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*******************************************
** interplanetary TCM trajectory optimization
** n-body heliocentric motion
** Earth-to-Mars data file - e2m_tcm.in
** May 8, 2012
*******************************************
```

The first program input is the Julian date, on the TDB time scale, at which to perform the TCM.

```
TDB julian date of TCM
2452799.264399034436792d0
```

The next six inputs are the components of the spacecraft's heliocentric orbital elements prior to the actual trajectory correction maneuver. These coordinates are defined relative to the Earth mean equator and equinox of J2000 system (EME2000) at the time specified in the previous input.

```
************************************************
heliocentric EME2000 orbital elements prior to TCM
************************************************

semimajor axis (kilometers)
190725765.750D0

orbital eccentricity (non-dimensional)
0.204056802425D0

orbital inclination (degrees)
23.4926769446D0

argument of perihelion (degrees)
253.488459798D0

right ascension of the ascending node (degrees)
0.463121032996D0

true anomaly (degrees)
3.94861259377D0
```

The next three inputs are the user's initial guess for the components of the TCM delta-v vector. These values should be provided in the units of meters per second. If an initial guess is not available, the user should input 0 for each component.

```
************************************************************
initial guess and bounds for heliocentric TCM delta-v vector
************************************************************

x-component of TCM velocity vector (meters/second)
0.0

y-component of TCM velocity vector (meters/second)
0.0

z-component of TCM velocity vector (meters/second)
0.0
```

Lower and upper bounds for each component of the TCM delta-v vector are defined by the next two user inputs. The units for these two numbers are also meters per second.

```
lower bound for TCM delta-v components (meters/second)
-100.0

upper bound for TCM delta-v components (meters/second)
+100.0
```

The next integer input allows the user to specify the type of final orbit targeting at Mars.

```
*****************************
final orbit targeting options
*****************************
 1 = user-defined flight path angle, radius and orbital inclination
 2 = user-defined B-plane coordinates
 3 = grazing flyby; user-defined b-plane angle
---------------------------------------------
1
```

The next set of inputs defines the desired characteristics of the encounter hyperbola at Mars. The orbital inclination and B-plane coordinates are defined with respect to the Mars mean equator and IAU node of epoch coordinate system.

```
*************************
final Mars-centered targets
*************************

flight path angle (degrees)
-2.0d0

periapsis radius (kilometers)
5000.0d0

orbital inclination (degrees)
60.0d0

user-defined b dot r target (kilometers)
-7889.908599155647607d0

user-defined b dot t target (kilometers)
4607.242716469171683d0

user-defined b-plane angle (degrees)
-60.0d0
```

The next two inputs set the root-finding and truncation error tolerances used by the numerical methods that calculate the initial guess, predict the time of entrance into the Martian sphere-of-influence, and the time of entry interface at Mars. Smaller values may help improve the solution of the corresponding problem at the expense of longer computation times.

```
*********************************************
root-finding and integration algorithm control
*********************************************

root-finding tolerance
1.0d-8

RKF7(8) truncation error tolerance
1.0d-12
```

The next input is the user-defined value for the radius of the sphere-of-influence of Mars.

```
-----------------------------------------------
radius of Mars sphere-of-influence (kilometers)
-----------------------------------------------
150000.0
```

This next input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by the *Sparse Optimization Suite* software. Options 2 and 3 allow the user to specify either the number of nodes (option 2) or time step size of the data file (option 3).

```
*********************************************
* type of comma-delimited solution data file *
*********************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this next input defines either the number of data points (option 2) or the time step size of the data output in the solution file (option 3).

```
number of user-defined nodes or print step size in solution data file
1
```

The name of the solution data file is defined in this next line.

```
name of solution output file
e2m_tcm.csv
```

The next series of program inputs are algorithm control options and parameters for the software. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the separated trapezoidal method is recommended.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next input defines the relative error in the objective function.

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

This integer input defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input.

```
**************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the *Sparse Optimization Suite* differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
***************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

Optimal control solution

The following is the program output created by the `itcm_sos` simulation for this example.

```
 program itcm_sos
 ================

 input data file ==> e2m_tcm.in

 user-defined fpa, radius and orbital inclination

 TCM delta-v vector and magnitude
 (heliocentric EME2000)
 ----------------------

 delta-vx             2.86350904748275      meters/second
 delta-vy             19.7001824350835      meters/second
 delta-vz            -2.66101349730156      meters/second

 deltav               20.0842690898477      meters/second

 TCM delta-v inertial pointing angles
 (heliocentric EME2000)
 ----------------------

 right ascension      81.7297238627486      degrees
 declination         -7.61364380952683      degrees

 TCM delta-v orbit-plane pointing angles

 pitch               -59.1104234734653      degrees
 yaw                  265.879855271489      degrees

 time and conditions prior to TCM maneuver
 (heliocentric EME2000)
 ----------------------

 calendar date        June  8, 2003

 TDB time             18:20:44.077

 TDB Julian date      2452799.26439903

      sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.190725765750D+09    0.204056802425D+00    0.234926769446D+02    0.253488459798D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.463121032996D+00    0.394861259377D+01    0.257437072392D+03    0.126193097746D+05

      rx (km)                ry (km)              rz (km)              rmag (km)
 -.319331575699D+08    -.136207676243D+09    -.590899587841D+08    0.151867971768D+09
      vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.316260605833D+02    -.655290816096D+01    -.295930903551D+01    0.324330976528D+02

 time and conditions after TCM maneuver
 (heliocentric EME2000)
 ----------------------
 calendar date        June  8, 2003
```

TDB time 18:20:44.077

TDB Julian date 2452799.26439903

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.190709071567D+09	0.203958745431D+00	0.234966353039D+02	0.253625947441D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.507601955901D+00	0.377033161794D+01	0.257396279059D+03	0.126176529628D+05

rx (km)	ry (km)	rz (km)	rmag (km)
-.319331575699D+08	-.136207676243D+09	-.590899587841D+08	0.151867971768D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.316289240924D+02	-.653320797852D+01	-.296197004901D+01	0.324321586131D+02

time and conditions at Mars SOI
(heliocentric EME2000)
---

calendar date December 23, 2003

TDB time 12:12:01.852

TDB Julian date 2452997.00835477

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.187247875806D+09	0.200438765484D+00	0.234405112944D+02	0.252515275783D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.349747173989D+00	0.153380415508D+03	0.458956912906D+02	0.122757179103D+05

rx (km)	ry (km)	rz (km)	rmag (km)
0.151506379502D+09	0.145182262180D+09	0.625457300247D+08	0.218961276544D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.144142894220D+02	0.157622522163D+02	0.687219371498D+01	0.224376331400D+02

time and conditions at Mars SOI
(Mars-centered mean equator and IAU node of epoch)
---

calendar date December 23, 2003

TDB time 12:12:01.852

TDB Julian date 2452997.00835477

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.584702528676D+04	0.185439805366D+01	0.599912397782D+02	0.113996849370D+03

raan (deg)	true anomaly (deg)	arglat (deg)
0.105662975108D+03	0.240791169623D+03	0.354788018992D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.337675013536D+05	0.145672670795D+06	-.117995275907D+05	0.150000000052D+06

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.548429658956D+00	-.273194181989D+01	0.362737401110D+00	0.280995722747D+01

B-plane coordinates at Mars SOI
(Mars-centered mean equator and IAU node of epoch)
---

b-magnitude	9131.08457768479	kilometers
b dot r	-7884.33839198771	kilometers
b dot t	4605.85645515161	kilometers

```
theta                 300.292529567016      degrees
v-infinity            2706.43946990651      meters/second
r-periapsis           4995.68702468775      kilometers
decl-asymptote        7.47149058794170      degrees
rasc-asymptote        281.318997229982      degrees

flight path angle    -86.6387448063388      degrees

time and conditions at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
--------------------------------------------------

calendar date         December 24, 2003

TDB time              02:07:38.158

TDB Julian date       2452997.58863608

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.585291050016D+04    0.185347485194D+01    0.600000000635D+02    0.113993614818D+03

    raan (deg)          true anomaly (deg)       arglat (deg)
 0.105636717750D+03    0.356921101065D+03    0.110914715883D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.176775951944D+04    -.234827585865D+04    0.404482714855D+04    0.499999998885D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.215455686466D+01    -.412635750164D+01    -.166729072802D+01    0.494457277084D+01

B-plane coordinates at Mars entry interface
(Mars-centered mean equator and equinox of date)
------------------------------------------------

b-magnitude           9133.85835280744      kilometers
b dot r              -7887.32724447201      kilometers
b dot t               4606.23895903804      kilometers
theta                 300.285144200537      degrees
v-infinity            2705.07843937188      meters/second
r-periapsis           4995.31192252423      kilometers
decl-asymptote        7.49072940722751      degrees
rasc-asymptote        281.282944565443      degrees

flight path angle    -2.00000000002416      degrees

time and conditions of Mars at entry interface
(heliocentric EME2000)
----------------------

calendar date         December 24, 2003

TDB time              02:07:38.158

TDB Julian date       2452997.58863608
     sma (au)              eccentricity       inclination (deg)       argper (deg)
 0.152368043785D+01    0.935420837075D-01    0.246772248956D+02    0.332979294518D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (days)
 0.337165819738D+01    0.704475973849D+02    0.434268919026D+02    0.686972361210D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.150787801633D+09    0.145972773409D+09    0.628789377224D+08    0.219086221628D+09

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 -.166518297346D+02    0.168787718592D+02    0.819178961924D+01    0.250854896335D+02

transfer time         198.324237048160      days
```

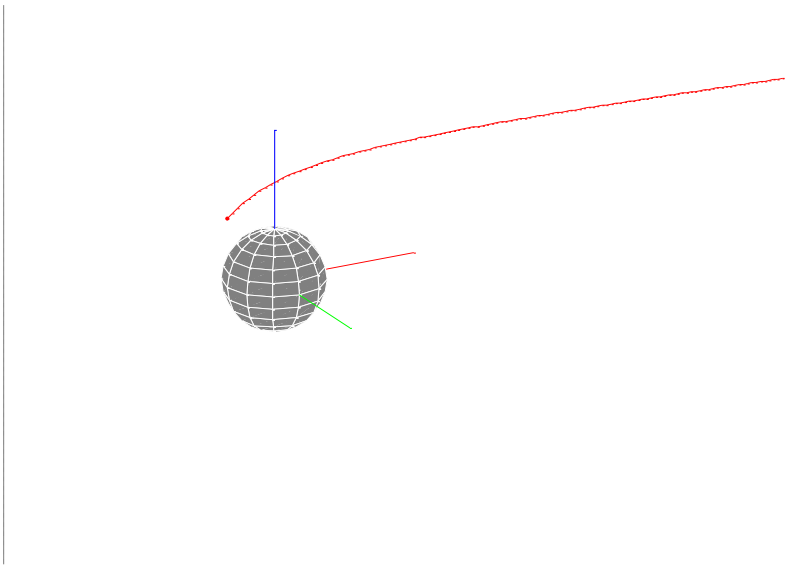
The `itcm_sos` software will create two comma-separated-variable (csv) output files. The first file has the name specified by the user and contains the heliocentric state vectors of the spacecraft and the Earth and Mars. The second file is named `soiorb.csv` and contains the Mars-centered state vector at the sphere-of-influence.

The following is the `soiorb.csv` data for this example.

```
0.2452997008354774D+07, -.1768303167878195D+04, -.2351415357895747D+04,
0.4044355626544280D+04, 0.2155870496202545D+01, -.4126374034966497D+01, -
.1667933926494574D+01,
```

The `itcm_sos` software package also includes a MATLAB script named `mplot.m` that can be used to create trajectory graphic displays using the `soiorb.csv` data file. The interactive graphic features of MATLAB allow the user to rotate and zoom the displays. These capabilities allow the user to interactively find the best viewpoint as well as verify basic orbital geometry of the hyperbolic encounter trajectory at Mars.

The following plot is a "zoomed" view of the encounter trajectory at Mars. This display is labeled with a Mars centered, inertial coordinate system. The x-axis of this areocentric system is red, the y-axis green and the z-axis blue. The small red dot is the location of the entry interface.



Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* software can be verified by numerically integrating the orbital equations of motion with the optimal control solution and the initial heliocentric conditions provided by the user. This is equivalent to solving an initial value problem (IVP) that uses the optimal impulsive thrust vector solution.

This part of the `itcm_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion. The following is a display of the final solution computed using this *explicit* numerical integration method.

```
=======================================
verification of optimal control solution
=======================================
```

time and conditions after TCM maneuver
(heliocentric EME2000)
---

calendar date June 8, 2003

TDB time 18:20:44.077

TDB Julian date 2452799.26439903

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.190709071567D+09	0.203958745431D+00	0.234966353039D+02	0.253625947441D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.507601955901D+00	0.377033161794D+01	0.257396279059D+03	0.126176529628D+05

rx (km)	ry (km)	rz (km)	rmag (km)
-.319331575699D+08	-.136207676243D+09	-.590899587841D+08	0.151867971768D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.316289240924D+02	-.653320797852D+01	-.296197004901D+01	0.324321586131D+02

time and conditions at Mars sphere-of-influence
(Mars-centered mean equator and IAU node of epoch)
---

calendar date December 23, 2003

TDB time 12:12:01.852

TDB Julian date 2452997.00835477

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.584702575071D+04	0.185441779440D+01	0.599953756038D+02	0.113996819994D+03

raan (deg)	true anomaly (deg)	arglat (deg)
0.105662253432D+03	0.240791638892D+03	0.354788458886D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.337667963805D+05	0.145671644140D+06	-.117989317052D+05	0.149998797440D+06

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.548429795002D+00	-.273194252894D+01	0.362737364915D+00	0.280995793872D+01

B-plane coordinates at Mars sphere-of-influence
(Mars-centered mean equator and IAU node of epoch)
---

b-magnitude	9131.22236299615	kilometers
b dot r	-7884.79362085750	kilometers
b dot t	4605.35030143918	kilometers
theta	300.288346415312	degrees
v-infinity	2706.43936253276	meters/second
r-periapsis	4995.80284573990	kilometers
decl-asymptote	7.47149194895923	degrees
rasc-asymptote	281.319000130777	degrees
flight path angle	-86.6386680328833	degrees

time and conditions at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
---

calendar date December 24, 2003

TDB time 02:07:38.158

TDB Julian date 2452997.58863608

```
     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.584304937602D+04    0.185515028038D+01    0.599817961400D+02    0.113992445290D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.105639013468D+03    0.356950775849D+03    0.110943221140D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.176830316788D+04    -.235141535790D+04    0.404435562654D+04    0.500128610588D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.215587049620D+01    -.412637403497D+01    -.166793392649D+01    0.494537600754D+01

B-plane coordinates at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
-------------------------------------------------

b-magnitude          9130.09430947819      kilometers
b dot r             -7882.77552850412      kilometers
b dot t              4606.56836128398      kilometers
theta                300.301334295321      degrees
v-infinity           2707.36011349352      meters/second
r-periapsis          4996.68531218008      kilometers
decl-asymptote       7.46173444525707      degrees
rasc-asymptote       281.299129226053      degrees

flight path angle   -1.98134902376417      degrees

time and conditions of Mars at entry interface
(heliocentric EME2000)
----------------------

calendar date        December 24, 2003

TDB time             02:07:38.158

TDB Julian date      2452997.58863608

     sma (au)              eccentricity       inclination (deg)       argper (deg)
 0.152368043785D+01    0.935420837075D-01    0.246772248956D+02    0.332979294518D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.337165819738D+01    0.704475973849D+02    0.434268919026D+02    0.686972361210D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.150787801633D+09    0.145972773409D+09    0.628789377224D+08    0.219086221628D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.166518297346D+02    0.168787718592D+02    0.819178961924D+01    0.250854896335D+02

transfer time        198.324237048160      days
```

Problem setup

This section provides additional details about the software implementation. For good scaling during the optimization, the time unit used in all heliocentric calculations is days, position is expressed in astronomical units, and the velocity and TCM delta-v unit is astronomical units per day. The areocentric or Mars-centered motion is modeled using seconds, kilometers, and kilometers per second.

During the solution of this optimal control problem, the `itcm_sos` software models the heliocentric motion *implicitly* using collocation and the Mars-centered orbital motion *explicitly* using numerical integration of the equations of motion.

Performance index – minimize TCM delta-v

The objective function or performance index *J* for this simulation is the scalar magnitude of the impulsive TCM delta-v vector. For this classic trajectory optimization problem, this index is simply $J = \Delta V$. The value of the `maxmin` indicator in *SOS* tells the software whether the user is minimizing or maximizing the performance index.

The components of the impulsive TCM delta-v are the optimization parameters used by the Sparse Optimization Suite to solve this trajectory problem.

Point functions – position and velocity vector "matching" at the TCM

The optimal solution must satisfy the following state vector boundary conditions (equality constraints) at the initial time defined by the user

$$\mathbf{r}_{s/c}^{-}\left(t_{TCM}\right) - \mathbf{r}_{s/c}^{+}\left(t_{TCM}\right) = 0$$

$$\mathbf{v}_{s/c}^{+}\left(t_{TCM}\right) - \left\{\mathbf{v}_{s/c}^{-}\left(t_{TCM}\right) + \Delta\mathbf{v}\left(t_{TCM}\right)\right\} = 0$$

where $\mathbf{r}_{s/c}^{-}$ and $\mathbf{v}_{s/c}^{-}$ are the heliocentric position and velocity vectors of the spacecraft prior to the TCM maneuver time t_{TCM}, $\mathbf{r}_{s/c}^{+}$ and $\mathbf{v}_{s/c}^{+}$ are the heliocentric position and velocity vectors of the spacecraft after the TCM delta-v is applied, and $\Delta\mathbf{v}$ is the *impulsive* TCM delta-v vector.

Point functions – SOI distance and final orbit targets at encounter

At the user-defined sphere-of-influence of Mars, the point function enforced by the software is given by $r_{SOI_p} - r_{SOI_u} = 0$ where r_{SOI_p} is the SOI value predicted by the software and r_{SOI_u} is the SOI value defined by the user.

Targeting to a Mars-centered flight path angle, periapsis radius and orbital inclination

For user-defined flight path angle, periapsis radius and orbital inclination targets at the Martian entry interface, the following point functions or equality constraints are enforced

$$\gamma_u - \gamma_p = 0 \qquad r_u - r_p = 0 \qquad \cos i_u - \hat{\mathbf{h}}_z = 0$$

where γ_u, r_u and i_u are the user-defined entry interface flight path angle, periapsis radius and Mars centered orbital inclination of the encounter hyperbola, respectively. In these equations, γ_p and r_p represent the flight path angle and periapsis radius predicted by the software. Note that $\hat{\mathbf{h}}_z$ is the z-component of the predicted unit angular momentum vector. These orbital characteristics are determined from the spacecraft's state vector at the entry interface relative to Mars.

Targeting to a Mars-centered grazing flyby

The general expression for the periapsis radius of an encounter hyperbola at Mars is given by

$$\tilde{r}_p = \frac{1}{\tilde{v}_\infty^2}\left(\sqrt{1 + \tilde{b}_\infty^2 \tilde{v}_\infty^4} - 1\right)$$

where the *normalized* quantities are

$$\tilde{r}_p = \text{ normalized periapsis radius} = r_p / r_m$$
$$\tilde{b}_\infty = \text{ normalized b-plane magnitude} = b_\infty / r_m$$
$$\tilde{v}_\infty = \text{ normalized v-infinity speed} = v_\infty / v_{lc}$$
$$v_{lc} = \text{ local circular speed at Mars} = \sqrt{\mu_m / r_m}$$
$$r_m = \text{radius of Mars}$$
$$\mu_m = \text{gravitational constant of Mars}$$

For a grazing flyby, $\tilde{r}_p = 1$ and the normalized B-plane distance is equal to $\tilde{b}_\infty = \sqrt{1 + 2/\tilde{v}_\infty^2}$. The required B-plane equality constraints are computed from $\mathbf{B} \bullet \mathbf{T} = b_\infty \cos\theta$ and $\mathbf{B} \bullet \mathbf{R} = b_\infty \sin\theta$, where θ is the user-defined B-plane angle of the grazing trajectory. Please note the B-plane angle is measured positive clockwise from the **T** axis of the B-plane coordinate system. The two equality constraints for this program option are simply the difference between the predicted and required $\mathbf{B} \bullet \mathbf{T}$ and $\mathbf{B} \bullet \mathbf{R}$ components.

Targeting to user-defined B-plane coordinates

For this program option, the two equality constraints are simply the difference between the predicted and user-defined $\mathbf{B} \bullet \mathbf{T}$ and $\mathbf{B} \bullet \mathbf{R}$ components. The B-plane constraints for this and the previous option are scaled by dividing by the equatorial radius of Mars.

Creating an initial guess

An initial guess for the heliocentric transfer trajectory is created by numerically integrating the heliocentric equations of motion using the user's input for the initial time and state of the spacecraft and the initial guess for the TCM delta-v vector. The position and velocity vectors at each grid point are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` which instructs the software to use the built-in Dormand-Prince solution method.

The final time for the heliocentric phase is estimated by searching for the instance at which the spacecraft reaches the user-defined sphere-of-influence distance at Mars. The numerical technique used to estimate the SOI consists of Brent's one-dimensional root-finder imbedded within a Runge-Kutta-Fehlberg 7(8) integrator. If an initial guess for the TCM maneuver is not available, a larger SOI radius should be used.

Technical discussion

Please consult the following Appendices for a technical summary of the equations of motion, B-plane orbital mechanics, and trajectory and time systems implemented in this computer program.

- *Appendix C – Cartesian Equations of Motion*
- *Appendix D – B-Plane Geometry, Coordinates and Targeting*
- *Appendix G – Aerospace Trajectory Coordinates and Time Systems*

In this computer program the heliocentric coordinates of the planets are based on the JPL Development Ephemeris DE421. These coordinates are provided in the Earth mean equator and equinox of J2000 coordinate system (EME2000).

Time from the Martian SOI to entry interface

The elapsed time from the Martian SOI until the entry interface is determined by an algorithm that includes a one-dimensional root-finder embedded within a Runge-Kutta-Fehlberg 7(8) numerical integration method. This technique searches for the time at which the predicted Mars-centered flight path angle of the spacecraft γ_p and the user-defined entry interface flight path angle γ_u is zero.

This mission constraint is computed as

$$\gamma_p - \gamma_u = \sin^{-1}\left(\frac{\mathbf{r}\bullet\mathbf{v}}{|\mathbf{r}\bullet\mathbf{v}|}\right) - \gamma_u = 0$$

where **r** and **v** are the Mars-centered position and velocity vectors, respectively.

Impulsive maneuver modeling

In the `itcm_sos` computer program, the components of the inertial unit thrust vector can be defined in terms of the right ascension α and the declination angle δ as follows

$$u_{T_{ECI_x}} = \cos\alpha\cos\delta \qquad u_{T_{ECI_y}} = \sin\alpha\cos\delta \qquad u_{T_{ECI_z}} = \sin\delta$$

The right ascension and declination angles can be determined from the components of the ECI unit thrust vector according to

$$\alpha = \tan^{-1}\left(u_{T_{ECI_y}}, u_{T_{ECI_x}}\right) \qquad \delta = \sin^{-1}\left(u_{T_{ECI_z}}\right)$$

where the inverse tangent calculation is a four-quadrant operation.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as follows:

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

Finally, the pitch and yaw angles can be determined from the components of the RTN unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

The pitch angle is positive above the “local horizontal” and the yaw angle is positive in the direction of the angular momentum vector.

Additional program examples

This section provides typical output from the `itcm_sos` computer program for the two other types of user-defined final orbit targeting options.

The first example illustrates the solution for the “user-defined B-plane coordinates” program option.

```
program itcm_sos
================

input data file ==> e2m_tcm.in

user-defined B-plane coordinates

TCM delta-v vector and magnitude
(heliocentric EME2000)
----------------------

delta-vx             2.86319939603731      meters/second
delta-vy             19.6995282017581      meters/second
delta-vz            -2.66239935347611      meters/second

deltav               20.0837668894737      meters/second

TCM delta-v inertial pointing angles
(heliocentric EME2000)
----------------------

right ascension      81.7303349840741      degrees
declination         -7.61782413127838      degrees

TCM delta-v orbit-plane pointing angles

pitch               -59.1061997378721      degrees
yaw                  265.879945402545      degrees

time and conditions prior to TCM maneuver
(heliocentric EME2000)
----------------------

calendar date        June  8, 2003

TDB time             18:20:44.077

TDB Julian date      2452799.26439903

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.190725765750D+09    0.204056802425D+00    0.234926769446D+02    0.253488459798D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.463121032996D+00    0.394861259377D+01    0.257437072392D+03    0.126193097746D+05

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.319331575699D+08    -.136207676243D+09    -.590899587841D+08    0.151867971768D+09

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.316260605833D+02    -.655290816096D+01    -.295930903551D+01    0.324330976528D+02

time and conditions after TCM maneuver
(heliocentric EME2000)
----------------------

calendar date        June  8, 2003

TDB time             18:20:44.077

TDB Julian date      2452799.26439903

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.190709070791D+09    0.203958744122D+00    0.234966356937D+02    0.253625930969D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.507606328913D+00    0.377034407992D+01    0.257396275048D+03    0.126176528859D+05

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.319331575699D+08    -.136207676243D+09    -.590899587841D+08    0.151867971768D+09

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.316289237827D+02    -.653320863275D+01    -.296197143486D+01    0.324321585694D+02
```

```
time and conditions at Mars SOI
(heliocentric EME2000)
----------------------

calendar date        December 23, 2003

TDB time             12:11:53.222

TDB Julian date      2452997.00825489

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.187247879371D+09    0.200438746485D+00    0.234405099417D+02    0.252515267845D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.349747285864D+00    0.153380366636D+03    0.458956344806D+02    0.122757182608D+05

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.151506522761D+09    0.145182113640D+09    0.625456610679D+08    0.218961257483D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.144142713323D+02    0.157622694803D+02    0.687220071981D+01    0.224376357922D+02

time and conditions at Mars SOI
(Mars-centered mean equator and IAU node of epoch)
--------------------------------------------------

calendar date        December 23, 2003

TDB time             12:11:53.222

TDB Julian date      2452997.00825489

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.584702316280D+04    0.185483277292D+01    0.599934925094D+02    0.113988421462D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.105662627673D+03    0.240800899993D+03    0.354789321455D+03
     rx (km)                ry (km)              rz (km)              rmag (km)
 -.337687834884D+05    0.145672590091D+06    -.117968549216D+05    0.150000000094D+06

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.548431763855D+00    -.273194184951D+01    0.362737662989D+00    0.280995770090D+01

B-plane coordinates at Mars SOI
(Mars-centered mean equator and IAU node of epoch)
--------------------------------------------------

b-magnitude          9134.09939400168      kilometers
b dot r             -7887.12464736178      kilometers
b dot t              4607.06376518497      kilometers
theta                300.290252792670      degrees
v-infinity           2706.43996147019      meters/second
r-periapsis          4998.22702360868      kilometers
decl-asymptote       7.47151398172520      degrees
rasc-asymptote       281.319031124716      degrees

flight path angle   -86.6376336996082      degrees

time and conditions at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
--------------------------------------------------

calendar date        December 24, 2003

TDB time             02:08:24.228

TDB Julian date      2452997.58916931

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.585238914353D+04    0.185398544607D+01    0.600026998845D+02    0.113983698032D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.105635154805D+03    0.943988629458D-11    0.113983698032D+03
```

```
     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.165099600309D+04    -.257162985432D+04    0.395467928201D+04    0.499785515330D+04

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 0.218557952117D+01    -.408034991477D+01    -.174091654773D+01    0.494538205769D+01
```

B-plane coordinates at Mars entry interface
(Mars-centered mean equator and equinox of date)

```
b-magnitude          9136.59360010819      kilometers
b dot r             -7889.90857959034      kilometers
b dot t              4607.24269159383      kilometers
theta                300.282416133723      degrees
v-infinity           2705.19892684709      meters/second
r-periapsis          4997.85515329978      kilometers
decl-asymptote       7.49076682019675      degrees
rasc-asymptote       281.281834267215      degrees

flight path angle   6.118383566213922E-012 degrees

time and conditions of Mars at entry interface
(heliocentric EME2000)
----------------------

calendar date        December 24, 2003

TDB time             02:08:24.228

TDB Julian date      2452997.58916931

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.152368043758D+01    0.935420835209D-01    0.246772248956D+02    0.332979294463D+03

    raan (deg)         true anomaly (deg)       arglat (deg)         period (days)
 0.337165819751D+01    0.704478985850D+02    0.434271930482D+02    0.686972361028D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 0.150787034465D+09    0.145973551028D+09    0.628793151252D+08    0.219086320051D+09

     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 -.166519174072D+02    0.168786869834D+02    0.819175305828D+01    0.250854787833D+02

transfer time        198.324770276900      days
```

```
========================================
```
verification of optimal control solution
```
========================================
```

```
time and conditions after TCM maneuver
(heliocentric EME2000)
----------------------

calendar date        June  8, 2003

TDB time             18:20:44.077

TDB Julian date      2452799.26439903

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.190709070791D+09    0.203958744122D+00    0.234966356937D+02    0.253625930969D+03

    raan (deg)         true anomaly (deg)       arglat (deg)         period (hrs)
 0.507606328913D+00    0.377034407992D+01    0.257396275048D+03    0.126176528859D+05

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.319331575699D+08    -.136207676243D+09    -.590899587841D+08    0.151867971768D+09
     vx (kps)              vy (kps)             vz (kps)             vmag (kps)
 0.316289237827D+02    -.653320863275D+01    -.296197143486D+01    0.324321585694D+02

time and conditions at Mars sphere-of-influence
(Mars-centered mean equator and IAU node of epoch)
--------------------------------------------------

calendar date        December 23, 2003
```

```
TDB time             12:11:53.222

TDB Julian date      2452997.00825489

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.584702362802D+04    0.185485195809D+01    0.599976005752D+02    0.113988400827D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.105661910874D+03    0.240801356489D+03    0.354789757316D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.337680811049D+05    0.145671568048D+06    -.117962651191D+05    0.149998803028D+06

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.548431898745D+00    -.273194255448D+01    0.362737627915D+00    0.280995840810D+01

B-plane coordinates at Mars sphere-of-influence
(Mars-centered mean equator and IAU node of epoch)
-------------------------------------------------

b-magnitude          9134.23331130423      kilometers
b dot r             -7887.57436946527      kilometers
b dot t              4606.55931813478      kilometers
theta                300.286097720332      degrees
v-infinity           2706.43985380146      meters/second
r-periapsis          4998.33959743101      kilometers
decl-asymptote       7.47151533485788      degrees
rasc-asymptote       281.319034008987      degrees

flight path angle   -86.6375584629909      degrees

time and conditions at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
-------------------------------------------------

calendar date        December 24, 2003

TDB time             02:08:24.228

TDB Julian date      2452997.58916931

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.584252256897D+04    0.185566113250D+01    0.599842372070D+02    0.113985732959D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.105638125635D+03    0.257396943962D-01    0.114011472653D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.165146923266D+04    -.257476031520D+04    0.395416961698D+04    0.499921980583D+04

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.218698975054D+01    -.408023503281D+01    -.174162041257D+01    0.494615848447D+01

B-plane coordinates at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
-------------------------------------------------

b-magnitude          9132.81449743411      kilometers
b dot r             -7885.33948200035      kilometers
b dot t              4607.57223471854      kilometers
theta                300.298656032703      degrees
v-infinity           2707.48216901316      meters/second
r-periapsis          4999.21947802138      kilometers
decl-asymptote       7.45899949361513      degrees
rasc-asymptote       281.300280791164      degrees

flight path angle   1.672612694781776E-002 degrees

time and conditions of Mars at entry interface
(heliocentric EME2000)
----------------------

calendar date        December 24, 2003
```

```
TDB time             02:08:24.228

TDB Julian date      2452997.58916931

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.152368043758D+01    0.935420835209D-01    0.246772248956D+02    0.332979294463D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.337165819751D+01    0.704478985850D+02    0.434271930482D+02    0.686972361028D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.150787034465D+09    0.145973551028D+09    0.628793151252D+08    0.219086320051D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.166519174072D+02    0.168786869834D+02    0.819175305828D+01    0.250854787833D+02

transfer time        198.324770276900      days
```

This next example illustrates the *SOS* solution for the "grazing flyby; user-defined B-plane angle" program option.

```
program itcm_sos
================

input data file ==> e2m_tcm.in

grazing flyby; user-defined B-plane angle

TCM delta-v vector and magnitude
(heliocentric EME2000)
----------------------

delta-vx              2.95580646628861      meters/second
delta-vy              20.0211618901866      meters/second
delta-vz             -2.71527341306197      meters/second

deltav                20.4195108904914      meters/second

TCM delta-v inertial pointing angles
(heliocentric EME2000)
----------------------

right ascension       81.6018526708799      degrees
declination          -7.64150893229803      degrees

TCM delta-v orbit-plane pointing angles

pitch                -59.1021477171684      degrees
yaw                   266.132390880707      degrees

time and conditions prior to TCM maneuver
(heliocentric EME2000)
----------------------

calendar date         June  8, 2003

TDB time              18:20:44.077

TDB Julian date       2452799.26439903

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 0.190725765750D+09    0.204056802425D+00    0.234926769446D+02    0.253488459798D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.463121032998D+00    0.394861259376D+01    0.257437072392D+03    0.126193097746D+05

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.319331575699D+08    -.136207676243D+09    -.590899587841D+08    0.151867971768D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.316260605833D+02    -.655290816096D+01    -.295930903551D+01    0.324330976528D+02

time and conditions after TCM maneuver
```

(heliocentric EME2000)
---

calendar date June 8, 2003

TDB time 18:20:44.077

TDB Julian date 2452799.26439903

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.190709610361D+09	0.203960529295D+00	0.234967036972D+02	0.253628262877D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.508369137299D+00	0.376731261224D+01	0.257395575490D+03	0.126177064343D+05

rx (km)	ry (km)	rz (km)	rmag (km)
-.319331575699D+08	-.136207676243D+09	-.590899587841D+08	0.151867971768D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.316290163898D+02	-.653288699907D+01	-.296202430892D+01	0.324321889230D+02

time and conditions at Mars SOI
(heliocentric EME2000)
---

calendar date December 23, 2003

TDB time 12:38:48.109

TDB Julian date 2452997.02694571

sma (km)	eccentricity	inclination (deg)	argper (deg)
0.187245719365D+09	0.200450404393D+00	0.234416910616D+02	0.252511180671D+03

raan (deg)	true anomaly (deg)	arglat (deg)	period (hrs)
0.353303393335D+00	0.153391925554D+03	0.459031062247D+02	0.122755058508D+05

rx (km)	ry (km)	rz (km)	rmag (km)
0.151479684137D+09	0.145210998446D+09	0.625576886883D+08	0.218965278503D+09

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
-.144176306146D+02	0.157585863250D+02	0.687138617596D+01	0.224369575362D+02

time and conditions at Mars SOI
(Mars-centered mean equator and IAU node of epoch)
---

calendar date December 23, 2003

TDB time 12:38:48.109

TDB Julian date 2452997.02694571

sma (km)	eccentricity	inclination (deg)	argper (deg)
-.584753723634D+04	0.158081701577D+01	0.602740922693D+02	0.120629580715D+03

raan (deg)	true anomaly (deg)	arglat (deg)
0.105606431802D+03	0.233443244170D+03	0.354072824886D+03

rx (km)	ry (km)	rz (km)	rmag (km)
-.327410572398D+05	0.145763800042D+06	-.134513233511D+05	0.150000001105D+06

vx (kps)	vy (kps)	vz (kps)	vmag (kps)
0.547720380833D+00	-.273189470329D+01	0.363279570246D+00	0.280984311512D+01

B-plane coordinates at Mars SOI
(Mars-centered mean equator and IAU node of epoch)
---

b-magnitude	7159.31165492607	kilometers
b dot r	-6199.74416616878	kilometers
b dot t	3580.35133002567	kilometers
theta	300.006427373413	degrees
v-infinity	2706.32099345972	meters/second
r-periapsis	3396.34912723837	kilometers
decl-asymptote	7.47129101837854	degrees

```
rasc-asymptote       281.312039235815      degrees

flight path angle   -87.3651678453474      degrees

time and conditions at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
-------------------------------------------------

calendar date        December 24, 2003

TDB time             02:30:12.997

TDB Julian date      2452997.60431710

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.586083603486D+04    0.157947188753D+01    0.602843087823D+02    0.120610993913D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.105538483942D+03    0.305333249420D-12    0.120610993913D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.932650560798D+03    -.205428844330D+04    0.253853629776D+04    0.339618971965D+04

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.270194114266D+01    -.434355754822D+01    -.252230120082D+01    0.570341840134D+01
```

B-plane coordinates at Mars entry interface

```
(Mars-centered mean equator and equinox of date)
------------------------------------------------

b-magnitude          7165.41183688191      kilometers
b dot r             -6205.42877860098      kilometers
b dot t              3582.70574647683      kilometers
theta                299.999998412226      degrees
v-infinity           2703.24879887418      meters/second
r-periapsis          3396.18971965012      kilometers
decl-asymptote       7.52267827567214      degrees
rasc-asymptote       281.215950946848      degrees

flight path angle   1.829382595895058E-013 degrees

time and conditions of Mars at entry interface
(heliocentric EME2000)
----------------------

calendar date        December 24, 2003

TDB time             02:30:12.997

TDB Julian date      2452997.60431710

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.152368042992D+01    0.935420782206D-01    0.246772248972D+02    0.332979292914D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.337165820116D+01    0.704564548650D+02    0.434357477788D+02    0.686972355849D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.150765239328D+09    0.145995639747D+09    0.628900355554D+08    0.219089116085D+09

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 -.166544077449D+02    0.168762757198D+02    0.819071437576D+01    0.250851705493D+02

transfer time        198.339918062557      days

========================================
verification of optimal control solution
========================================

time and conditions after TCM maneuver
(heliocentric EME2000)
----------------------

calendar date        June  8, 2003
```

```
TDB time             18:20:44.077

TDB Julian date      2452799.26439903

     sma (km)            eccentricity       inclination (deg)       argper (deg)
 0.190709610361D+09    0.203960529295D+00    0.234967036972D+02    0.253628262877D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (hrs)
 0.508369137299D+00    0.376731261224D+01    0.257395575490D+03    0.126177064343D+05

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.319331575699D+08    -.136207676243D+09    -.590899587841D+08    0.151867971768D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.316290163898D+02    -.653288699907D+01    -.296202430892D+01    0.324321889230D+02

time and conditions at Mars sphere-of-influence
(Mars-centered mean equator and IAU node of epoch)
-------------------------------------------------

calendar date        December 23, 2003

TDB time             12:38:48.109

TDB Julian date      2452997.02694571

     sma (km)            eccentricity       inclination (deg)       argper (deg)
 -.584753764517D+04    0.158083739491D+01    0.602792488265D+02    0.120629421786D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.105605536266D+03    0.233443926995D+03    0.354073348781D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.327403659241D+05    0.145762745019D+06    -.134507194550D+05    0.149998770829D+06

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.547720524843D+00    -.273189544180D+01    0.363279541054D+00    0.280984385744D+01

B-plane coordinates at Mars sphere-of-influence
(Mars-centered mean equator and IAU node of epoch)
-------------------------------------------------

b-magnitude          7159.46602063614      kilometers
b dot r             -6200.20371898383      kilometers
b dot t              3579.86417951053      kilometers
theta                300.001212120693      degrees
v-infinity           2706.32089885404      meters/second
r-periapsis          3396.46853247520      kilometers
decl-asymptote       7.47129230001981      degrees
rasc-asymptote       281.312042151398      degrees

flight path angle   -87.3650901566508      degrees

time and conditions at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
-------------------------------------------------

calendar date        December 24, 2003

TDB time             02:30:12.997

TDB Julian date      2452997.60431710

     sma (km)            eccentricity       inclination (deg)       argper (deg)
 -.583079150857D+04    0.158274748840D+01    0.602550775360D+02    0.120629395635D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.105549108857D+03    0.236231844260D-01    0.120653018819D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.932799718350D+03    -.205772782893D+04    0.253795675613D+04    0.339787928397D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.270578341740D+01    -.434272721858D+01    -.252458920092D+01    0.570561953080D+01
```

```
B-plane coordinates at Mars entry interface
(Mars-centered mean equator and IAU node of epoch)
--------------------------------------------------

b-magnitude          7153.33710334291      kilometers
b dot r             -6193.60832320210      kilometers
b dot t              3579.02887007410      kilometers
theta                300.021832410397      degrees
v-infinity           2710.20441327616      meters/second
r-periapsis          3397.87910698875      kilometers
decl-asymptote       7.42062702840562      degrees
rasc-asymptote       281.280927562393      degrees

flight path angle   1.447665173567820E-002 degrees

time and conditions of Mars at entry interface
(heliocentric EME2000)
----------------------

calendar date        December 24, 2003

TDB time             02:30:12.997

TDB Julian date      2452997.60431710

     sma (au)             eccentricity       inclination (deg)       argper (deg)
 0.152368042992D+01    0.935420782206D-01    0.246772248972D+02    0.332979292914D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (days)
 0.337165820116D+01    0.704564548650D+02    0.434357477788D+02    0.686972355849D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.150765239328D+09    0.145995639747D+09    0.628900355554D+08    0.219089116085D+09

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 -.166544077449D+02    0.168762757198D+02    0.819071437576D+01    0.250851705493D+02

transfer time        198.339918062557      days
```

Contents of the simulation summary and csv files

This material is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `itcm_sos` software.

The simulation summary screen display contains the following information:

calendar date = calendar date of trajectory event

TDB time = Barycentric Dynamical Time of trajectory event

TDB julian date = Barycentric Dynamical Time julian date of trajectory event

sma (km) = semimajor axis in kilometers or astronomical units

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (days) = orbital period in hours or days

delta-vx = x-component of the impulsive TCM velocity vector in meters/second

delta-vy = y-component of the impulsive TCM velocity vector in meters/second

delta-vz = z-component of the impulsive TCM velocity vector in meters/second

deltav = scalar magnitude of the impulsive TCM delta-v in meters/seconds

transfer time = elapsed time from TCM to entry interface in days

The comma-separated-variable disk file is created by the `odeprt` subroutine and contains the following information:

time (days) = simulation time since launch in days

rs2sc-x (au) = x-component of spacecraft position vector in astronomical units

rs2sc-y (au) = y-component of spacecraft position vector in astronomical units

rs2sc-z (au) = z-component of spacecraft position vector in astronomical units

rs2sc-mag (au) = heliocentric radius magnitude of spacecraft in astronomical units

vs2sc-x (km/sec) = x-component of spacecraft velocity vector in kilometers per second

vs2sc-y (km/sec) = y-component of spacecraft velocity vector in kilometers per second

vs2sc-z (km/sec) = z-component of spacecraft velocity vector in kilometers per second

vs2sc-mag (km/sec) = heliocentric velocity of spacecraft in kilometers per second

rs2e-x (au) = x-component of Earth position vector in astronomical units

rs2e-y (au) = y-component of Earth position vector in astronomical units

rs2e-z (au) = z-component of Earth position vector in astronomical units

rs2e-mag (au) = heliocentric radius magnitude of Earth in astronomical units

rs2m-x (au) = x-component of Mars position vector in astronomical units

rs2m-y (au) = y-component of Mars position vector in astronomical units

rs2m-z (au) = z-component of Mars position vector in astronomical units

rs2m-mag (au) = heliocentric radius magnitude of Mars in astronomical units

sma (au) = transfer orbit semimajor axis in astronomical units

eccentricity = transfer orbit orbital eccentricity (non-dimensional)

inclination (deg) = transfer orbit orbital inclination in degrees

argper (deg) = transfer orbit argument of perigee in degrees

raan (deg) = transfer orbit right ascension of the ascending node in degrees

true anomaly (deg) = transfer orbit true anomaly in degrees

The heliocentric coordinates of the spacecraft are with respect to the Earth mean equator and equinox of J2000 coordinate system.

"Interplanetary Mission Design Handbook, Volume 1, Part 2", JPL Publication 82-43, September 15, 1983.

"Error Analysis of Multiple Planet Trajectories", F. M. Sturms, Jr., JPL Space Programs Summary, No. 37-27, Vol. IV.

"Optimal Interplanetary Orbit Transfers by Direct Transcription", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 42, No. 3, July-September 1994, pp. 247-268.

"Design and Optimization of Interplanetary Spacecraft Trajectories", Thomas T. McConaghy, PhD thesis, Purdue University, December 2004.

"A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem" by R. H. Gooding, *Celestial Mechanics and Dynamical Astronomy* **48:** 145-165, 1990.

CMATO 16 – Optimal Finite-Burn Earth Orbit-to-Interplanetary Injection

This *CMATO* application is a Fortran computer program named `hyper_sos` that uses the *Sparse Optimization Suite* (*SOS*) distributed by Applied Mathematical Analysis to solve the single-maneuver, finite-burn Earth orbit-to-interplanetary injection trajectory optimization problem. The software attempts to maximize the final spacecraft mass. Since this simulation involves a single propulsive maneuver, this is equivalent to minimizing the propellant mass required for the orbital maneuver. The `hyper_sos` software can also be used to solve this problem using a fixed thrust duration maneuver.

The important features of this scientific simulation are as follows:

- single, continuous thrust propulsive maneuver
- inertially fixed or variable attitude steering
- modified equinoctial equations of motion
- valid for circular and elliptical park orbits
- user-specified departure hyperbola targets (C3, RLA and DLA)
- user-specified final coast duration

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `hyper_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method.

The `hyper_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `hyper_sos` software is "data-driven" by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in `courier` font and all explanations are in times font. Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or change the number of lines. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. All program inputs and outputs are in an Earth-centered-inertial (ECI) coordinate system. Park orbit angular elements and the hyperbolic targets should be specified in the same coordinate system.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
*********************************************************************
** Optimal Finite-Burn Earth Orbit-to-Interplanetary Injection
** single phase, continuous-thrust maneuver with user-defined final coast
** variable and fixed inertial attitude steering
** input file ==> hyper1.in - February 14, 2011
*********************************************************************
```

The first input defines the type of steering to use during the solution process. Additional information about these steering options can be found in the Problem setup section.

```
type of steering
1 = fixed inertial attitude
2 = variable attitude
---------------------
2
```

The next two inputs allow the user to provide an initial guess for the right ascension and declination for the inertially fixed steering option.

```
initial guess for right ascension (degrees)
0.0d0

initial guess for declination (degrees)
0.0d0
```

The next three inputs define the initial mass, thrust magnitude and specific impulse of the propulsion system, respectively.

```
initial spacecraft mass (kilograms)
4000.0

thrust magnitude (newtons)
19840.0

specific impulse (seconds)
450.0
```

The next input defines the user's initial guess for the maneuver duration, in seconds.

```
initial guess for thrust duration (seconds)
550.0
```

The next two inputs define the lower and upper bounds for the thrust duration, also in seconds. For a fixed thrust duration mission, these two numbers should be identical.

```
lower bound for thrust duration (seconds)
1.0

upper bound for thrust duration (seconds)
1000.0
```

The next six user-defined inputs define the classical orbital elements of the initial park orbit.

```
*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
6563.34

orbital eccentricity (non-dimensional)
0.015

orbital inclination (degrees)
28.5

argument of perigee (degrees)
90.0

right ascension of the ascending node (degrees)
0.0

true anomaly (degrees)
145.0
```

This next integer input allows the user to define the type of initial park orbit constraints to use during the simulation. Option 1 will constrain the park orbit semimajor axis, eccentricity and orbital inclination to the values input by the user. All other elements are unconstrained. Option 2 will constrain all initial elements, and option 3 will constrain all initial elements except true longitude.

```
*******************************
initial orbit constraint options
*******************************
1 = constrain semimajor axis, eccentricity, and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-------------------------------------------
1
```

The next three inputs define the characteristics of the departure hyperbola. These inputs are the specific orbital energy (C3) in kilometers per second squared, and the right ascension (RLA) and declination (DLA) of the outgoing asymptote, respectively.

```
********************
* LAUNCH HYPERBOLA *
********************

specific orbital energy (C3; [km/sec]**2)
8.788564

right ascension of outgoing asymptote (RLA; degrees)
349.68004

declination of outgoing asymptote (DLA; degrees)
-6.666253
```

This next numeric input allows the user to specify a final fixed-duration orbit coast. For a simulation without a final coast, set this value to zero.

```
final coast duration (seconds)
100.0
```

This integer input specifies the type of gravity model to use during the simulation. Option 2 will include the contribution of the oblateness gravity coefficient (J_2) in the equations of motion.

```
************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate Earth
----------------
2
```

This next input defines the type of initial guess to use. Please see the technical discussion section for information about how the first option is modeled. Option 2 requires either a binary restart file created from a previous run using either initial guess option 1 or an updated binary restart file. This feature is described in the next two sections.

```
************************
* initial guess options *
************************
 1 = numerical integration
 2 = binary data file
---------------------
1
```

If the user elects to use a binary data file (option 3 above) for the initial guess, the following text input specifies the name of the file to use.

```
name of binary initial guess data file
hyper1.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess option 1, the software will create a binary restart file. For initial guess option 2, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*****************************
* binary restart file option *
*****************************

create/update binary data file (yes or no)
no
```

This next input specifies the type of comma-delimited or comma-separated-variable (CSV) solution data file to create. Option 1 will create a solution file at each collocation point or node determined by SOS. Options 2 and 3 allow the user to specify either the number of nodes or time step size used to create the data file.

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this next input defines either the number of data points or the time step size of the data written to the solution file.

```
number of user-defined nodes or print step size in solution data file
25
```

The name of the solution data file is defined in this next line.

```
name of solution output file
hyper1.csv
```

The next series of program inputs are algorithm control options and parameters for the *Sparse Optimization Suite*. The first input is an integer that specifies the type of collocation method to use during the solution process.

```
*******************************
* algorithm control parameters *
*******************************

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
---------------------------------
1
```

The next input is an integer that defines the number of grid points to use for the initial guess. If necessary, the *Sparse Optimization Suite* will use mesh refinement to change the number and distribution of the grid points.

```
number of grid points
25
```

The next input defines the relative error in the objective function (performance index).

```
relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the NLP algorithm is controlled with the following integer input. Additional information about these algorithm items can be found in the *Sparse Optimization Suite* user's manual.

```
**************************
sparse NLP iteration output
--------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the *Sparse Optimization Suite* optimal control algorithm is controlled with the following integer input. Note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the *Sparse Optimization Suite* differential equations algorithm is controlled with the following integer input. Note that option 5 will create lots of information.

```
***************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the `SOCOUT` character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of user inputs allow the reading and writing of configuration input files by the simulation. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `hyper_sos` software. Additional information about this type of file can be found in *Appendix J – Sparse Optimization Suite Configuration File.*

```
************************************
* optimal control configuration options
************************************

read an optimal control configuration file (yes or no)
no
```

```
name of optimal control configuration file
hyper1_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
hyper1_config1.txt
```

Optimal control solution

The following is the solution for this example. This simulation used an oblate Earth gravity model, a numerical integration initial guess, variable attitude steering, and a final 100 second coast. The program output includes the orbital characteristics at the beginning and end of the propulsive maneuver.

```
program hyper_sos
=================

input file ==> hyper1.in

numerical integration initial guess

variable attitude steering

oblate earth gravity model

------------------------
beginning of finite burn
------------------------

     sma (km)            eccentricity       inclination (deg)       argper (deg)
 0.656334000000D+04    0.150000000000D-01    0.285000000000D+02    0.195006107461D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.213742358980D+01    0.340195936757D+03    0.175202044219D+03    0.881956335064D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.646112461247D+04    0.234812204181D+03    0.258243682196D+03    0.647054540425D+04

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 -.362932694068D+00    -.694302305251D+01    -.375978161985D+01    0.790400196591D+01
```

We can see how the software has modified the angular elements of the park orbit.

```
------------------
end of finite burn
------------------

     sma (km)            eccentricity       inclination (deg)       argper (deg)
 -.453027509263D+05    0.114402245190D+01    0.284832207197D+02    0.194946919587D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.212549933630D+01    0.211148057289D+02    0.216061725316D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.533674174038D+04    -.370176984989D+04    -.189971873374D+04    0.676704098498D+04

     vx (kps)              vy (kps)              vz (kps)             vmag (kps)
 0.504708187457D+01    -.879765348372D+01    -.487168058542D+01    0.112518893522D+02
```

The following program output is the final spacecraft mass, the propellant mass consumed, the actual thrust duration for the maneuver, and the accumulated delta-v. The delta-v is computed using a cubic spline numerical method which uses all data points of the output file.

```
final mass              1764.41952920903      kilograms

propellant mass         2235.58047079097      kilograms

thrust duration         497.258056991095      seconds
                        8.28763428318491      minutes
delta-v                 3611.91380012394      meters/second
```

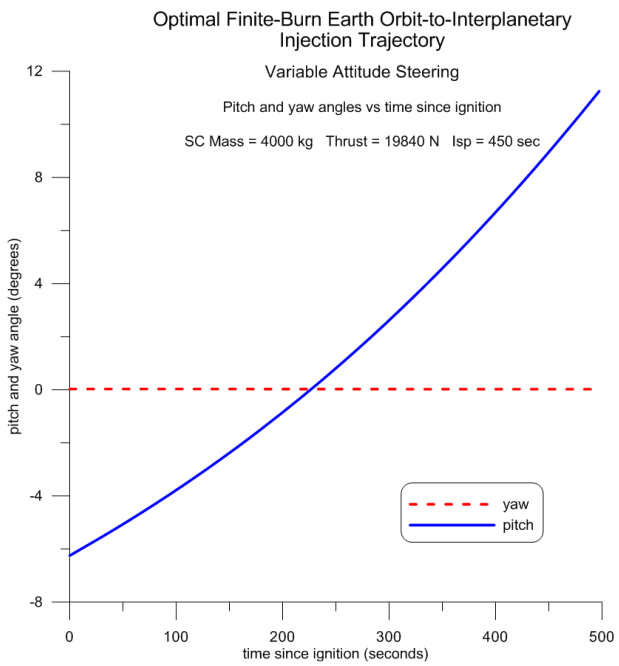
This section of the output file summarizes the classical orbital elements and Earth-centered-inertial (ECI) state vector at the end of the user-defined coast.

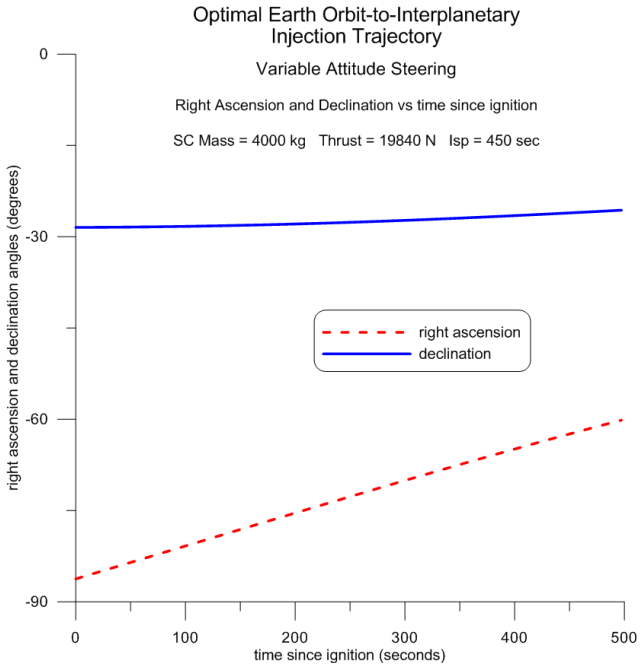
```
------------------
end of final coast
------------------

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.453544448786D+05    0.114386211719D+01    0.284806735260D+02    0.194956415150D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.212090855648D+01    0.301198658443D+02    0.225076280994D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.479993020204D+04    -.455676834369D+04    -.237406820531D+04    0.703133469023D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.566636372018D+01    -.829462754194D+01    -.461067843664D+01    0.110529127163D+02
```

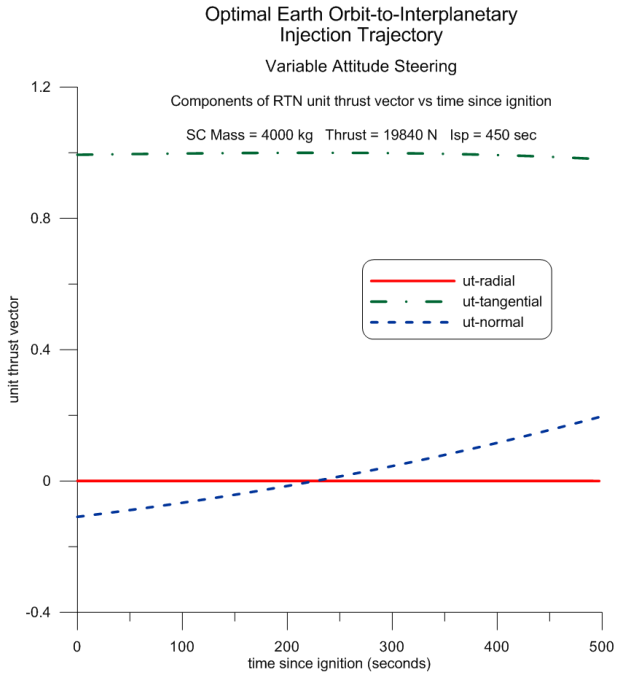
This final section of the program output displays the actual hyperbolic conditions computed by the software. From this display we can determine how well the software satisfies the user-defined C3, RLA and DLA targets.

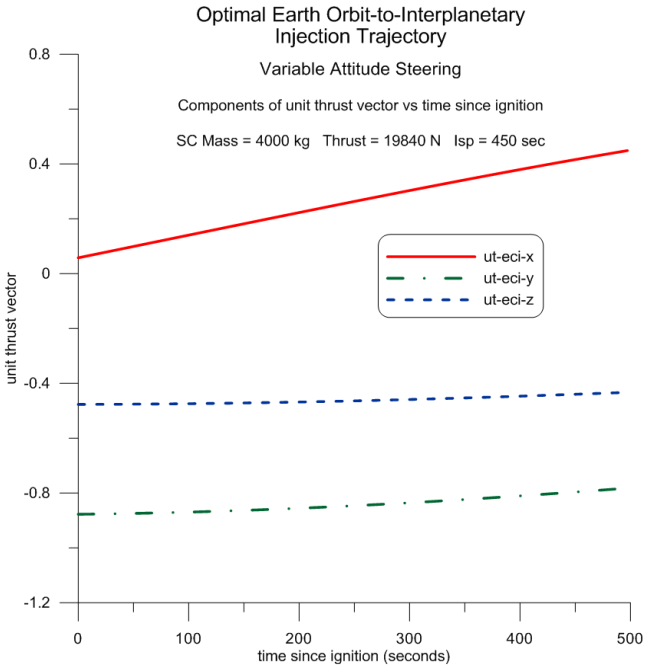
```
------------------
outgoing hyperbola
------------------

right ascension          349.680040000001      degrees

declination             -6.66625299999994      degrees

orbital energy           8.78856399999989      (km/sec)**2

final coast duration     100.000000000000      seconds
                         1.66666666666667      minutes
```

The following are plots of the behavior of the pitch and yaw angles, the inertial right ascension and declination angles, and the components of the ECI and RTN unit thrust vector during the maneuver.

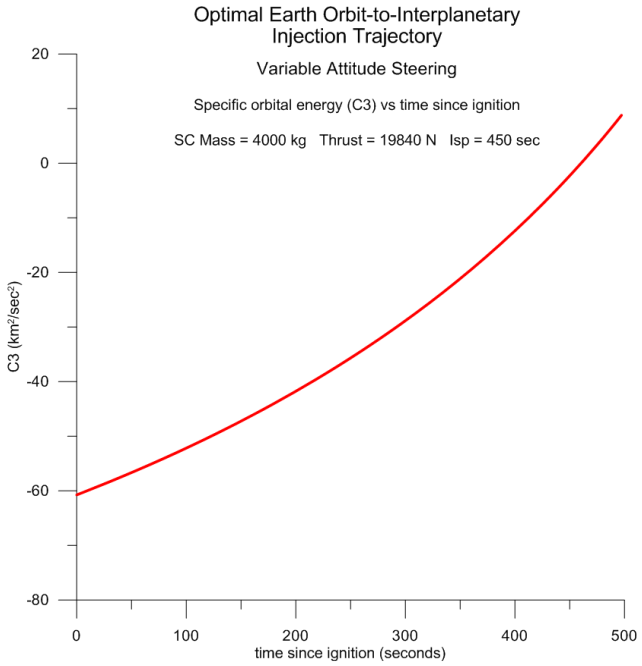


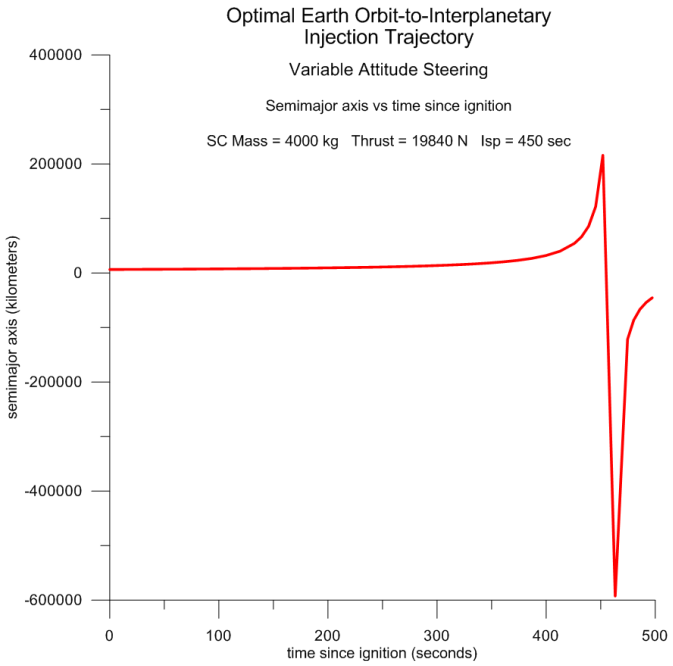




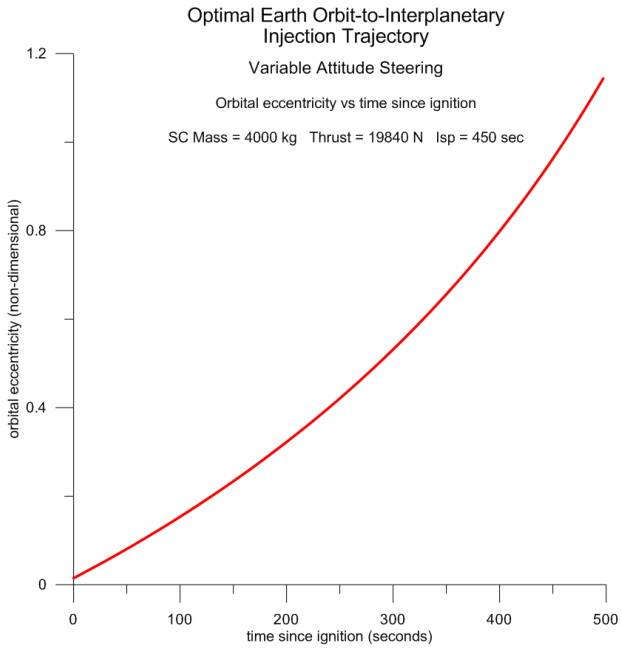


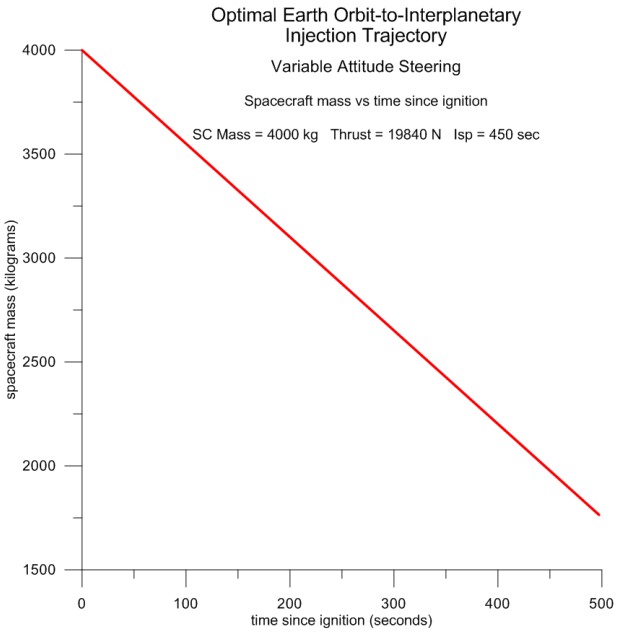
The following plots illustrate the evolution of the specific orbital energy (C3) and the semimajor axis of the spacecraft's orbit as a function of time since ignition. We can see the transition from an elliptical to a hyperbolic orbit, especially the plot of semimajor axis, as orbital energy is added to the trajectory.





These final two plots illustrate the time evolution of the orbital eccentricity of the transfer orbit and the spacecraft's instantaneous mass.

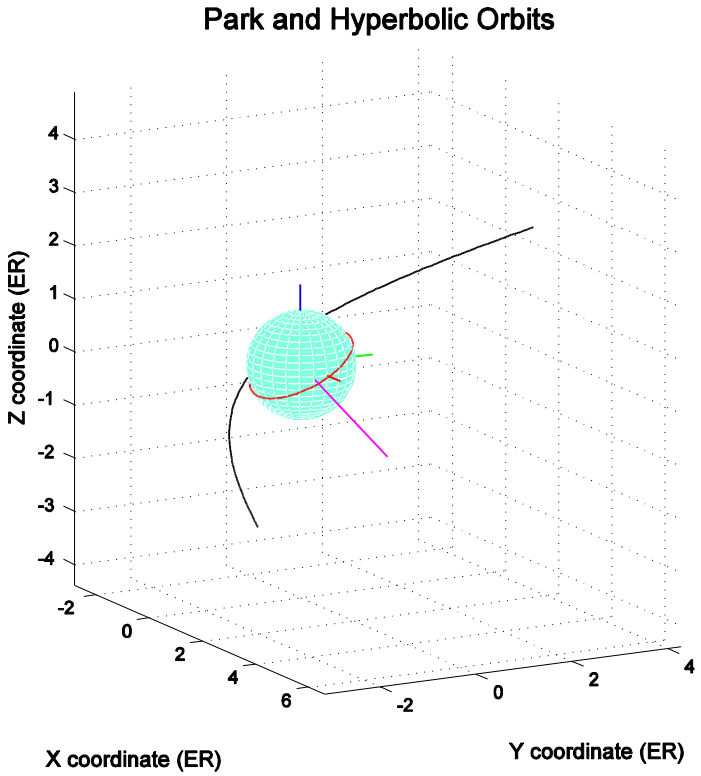




The `hyper_sos` computer program will also create an output file named `hyper_sos.dat`. This file contains the Earth-centered inertial position and velocity vectors of the park orbit and launch hyperbola at interplanetary injection, and the orientation of the departure asymptote. It is written as a single column of data.

The next plot is a typical display of the park and hyperbolic orbits for this example. This display is labeled with an Earth centered, inertial coordinate system. The x-axis of this system is red, the y-axis green and the z-axis blue. The outgoing asymptote is colored magenta, the park orbit trace is red, and the hyperbolic trajectory is black.

The fundamental plane of this right-handed, orthogonal coordinate system is the Earth's equator. The x-axis points in the direction of the Vernal Equinox, the z-axis is aligned with the Earth's spin axis, and the y-axis is advanced 90 degrees along the equator from the x-axis. As mentioned earlier, the classical orbital elements of the park orbit and the hyperbolic targets (C3, RLA and DLA) should be provided in the same ECI coordinate system, perhaps true-of-date or Earth mean equator and equinox of J2000.



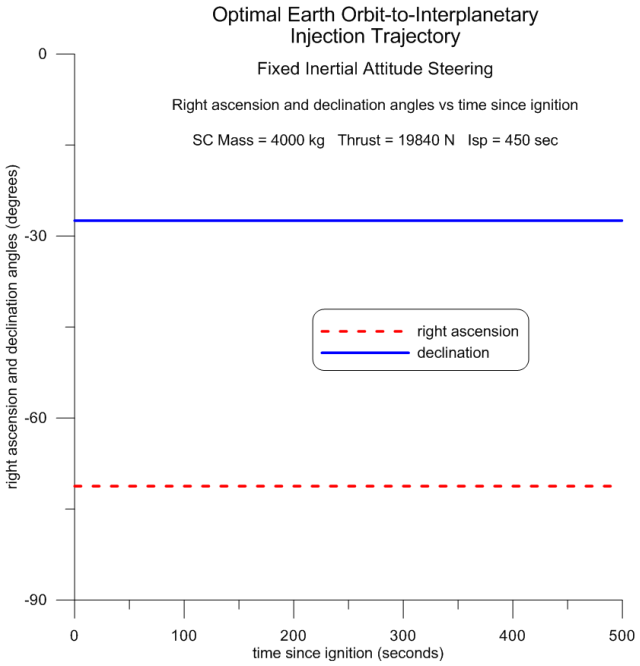
Fixed inertial attitude example

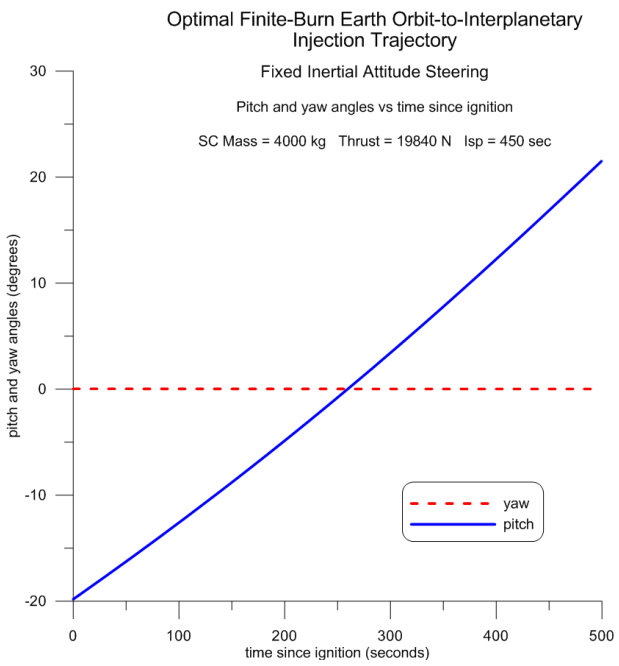
This section summarizes the output for this same example with fixed inertial attitude steering. The output includes the optimized right ascension and declination angles. The initial guess for the right ascension and declination angles were both set to zero.

```
------------------------
beginning of finite burn
------------------------

     sma (km)              eccentricity        inclination (deg)        argper (deg)
 0.656334000000D+04    0.150000000000D-01    0.285000000000D+02    0.194942230990D+03

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.213772655895D+01    0.339972859914D+03    0.174915090905D+03    0.881956335064D+02

     rx (km)                ry (km)               rz (km)               rmag (km)
 -.645952149379D+04    0.263240915368D+03    0.273654514483D+03    0.647067233767D+04

     vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 -.401992811987D+00    -.694155708635D+01    -.375819334287D+01    0.790384907537D+01

------------------
end of finite burn
------------------

     sma (km)              eccentricity        inclination (deg)        argper (deg)
 -.453006589465D+05    0.114267116923D+01    0.284827756905D+02    0.194823732303D+03
```

```
     raan (deg)         true anomaly (deg)       arglat (deg)
  0.212595426613D+01    0.214243278329D+02    0.216248060135D+03

      rx (km)                ry (km)              rz (km)             rmag (km)
  -.527858325139D+04    -.368582862394D+04    -.189218731497D+04    0.671037604176D+04

      vx (kps)               vy (kps)             vz (kps)            vmag (kps)
  0.507174069203D+01    -.883000610414D+01    -.488964831758D+01    0.112960268288D+02

final mass               1754.15735504550      kilograms

 propellant mass          2245.84264495450      kilograms

 thrust duration          499.540662719961      seconds
                          8.32567771199936      minutes

 delta-v                  3637.65698948535      meters/second

 inertial steering angles

 right ascension         -71.2198741186687      degrees

 declination             -27.4489611164256      degrees

 ------------------
 end of final coast
 ------------------

      sma (km)              eccentricity       inclination (deg)       argper (deg)
  -.453544448786D+05    0.114250588430D+01    0.284801507703D+02    0.194833467431D+03

     raan (deg)         true anomaly (deg)       arglat (deg)
  0.212118370349D+01    0.305275723761D+02    0.225361039807D+03

      rx (km)                ry (km)              rz (km)             rmag (km)
  -.473889037681D+04    -.454354842152D+04    -.236806227854D+04    0.697915705635D+04

      vx (kps)               vy (kps)             vz (kps)            vmag (kps)
  0.569856258595D+01    -.831667145066D+01    -.462318936301D+01    0.110911910745D+02

 ------------------
 outgoing hyperbola
 ------------------

 right ascension          349.680040000000      degrees

 declination             -6.66625299999999      degrees

 orbital energy           8.78856400000002      (km/sec)**2

 final coast duration     100.000000000000      seconds
                          1.66666666666667      minutes

 ----------------------------------------
 verification of optimal control solution
 ----------------------------------------

 right ascension          349.680040437411      degrees

 declination             -6.66625277125702      degrees

 orbital energy           8.78856365891086      (km/sec)**2

 final mass               1754.15735504558      kilograms

 propellant mass          2245.84264495442      kilograms

 delta-v                  3637.65513253191      meters/second
```

The following are plots of the behavior of the inertial right ascension and declination angles, and the evolution of the yaw and pitch angles during the propulsive maneuver.





Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* can be verified by numerically integrating the orbital equations of motion with the *SOS*-computed initial park orbit conditions and the optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the *SOS* optimal unit thrust vector solution. This part of the `hyper_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is a display of the final solution computed using this *explicit* numerical integration method. These results are for the variable attitude example.

```
----------------------------------------
verification of optimal control solution
----------------------------------------

right ascension          349.680039311870      degrees

declination             -6.66625334384853      degrees

orbital energy           8.78856386226052      (km/sec)**2


final mass               1764.41952921893      kilograms

propellant mass          2235.58047078107      kilograms

delta-v                  3611.91346050931      meters/second
```

A comparison of these results with the earlier optimal control solution exhibits excellent agreement. The delta-v was computed by including the thrust acceleration in the verification equations of motion.

Creating an Initial Guess

This section describes several methods that can be used to create an initial guess for `hyper_sos`. The *Sparse Optimization Suite* attempts to obtain problem feasibility before trying to solve the optimal

control problem. If the algorithm cannot find a feasible solution, the software will print warnings and the user will have to provide a better initial guess.

The software requires an initial guess for the thrust duration. The user should also provide lower and upper bounds for the total thrust duration. These three inputs should be in seconds.

For hyperbolic injection from circular and near-circular orbits, the delta-v magnitude can be estimated using the numerical technique described in *Appendix H – Impulsive Interplanetary Injection from a Circular Earth Orbit*.

From the user-defined initial spacecraft mass and delta-v estimate, the propellant mass required for the maneuver is given by the ideal rocket equation as

$$m_p = m_i \left(1 - e^{\frac{-\Delta V}{V_{ex}}} \right)$$

An estimate of the thrust duration of the maneuver can be determined from

$$t_d = \frac{g I_{sp} m_p}{F} = \frac{V_{ex} m_p}{F}$$

where

$F =$ thrust

$m_i =$ pre-maneuver spacecraft mass

$V_{ex} =$ exhaust velocity = $g I_{sp}$

$I_{sp} =$ specific impulse

$g =$ acceleration of gravity at Earth surface

Since a finite burn maneuver will require a thrust duration longer than this impulsive estimate, the user can increase this value by about 10% and use it for the initial guess required by `hyper_sos`.

For the numerical integration initial guess option with variable attitude steering, the software models the maneuver using tangential thrusting. For this case, the unit thrust vector in the modified equinoctial frame at all times is simply $\mathbf{u}_T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$. Please note that this type of steering method creates a *coplanar* initial guess that increases the spacecraft's orbital energy.

For the fixed inertial attitude steering option, the software uses the user-defined initial guess for right ascension and declination to create a unit thrust vector in the Earth-centered-inertial (ECI) coordinate system. The software then transforms this inertial unit thrust vector to the modified equinoctial frame and integrates the equations of motion in this system. For many simulations, the initial guesses for both angles can be set to zero. If the simulation does not converge with these simple guesses, the user can input "averaged" angles determined from the output of an optimized variable attitude case.

The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine for this aerospace trajectory optimization problem. These program options create an initial guess from the numerical integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* to construct an initial guess by solving an initial value problem (IVP) with a

linear control approximation. The `INIT(2) = 2` program option tells the program to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

Binary restart data files can also be used to initialize a `hyper_sos` simulation. A typical scenario is

1) create a binary restart file from a converged and optimized simulation
2) modify the original input file with slightly different spacecraft characteristics, propulsive properties or perhaps final mission targets and/or constraints
3) use the previously created binary restart file as the initial guess for the new simulation

This technique works well provided the two simulations are not dramatically different. Sometimes it is necessary to make successive small changes in the mission definition and run multiples simulations to eventually reach the final desired solution.

Problem setup

This section provides details about the software implementation within `hyper_sos`. It defines such things as point and path constraints (boundary conditions), bounds on the dynamic variables, and the performance index or objective function.

Point functions – initial orbit constraints

The software allows the user to select one of the following initial orbit constraint options.

1) constrain semimajor axis, eccentricity and inclination
2) constrain all initial orbital elements
3) option 2 with unconstrained true longitude

For all three options, the initial orbit inclination is constrained by enforcing

$$\sqrt{h^2 + k^2} = \tan\left(\frac{i}{2}\right)$$

where *i* is the park orbit inclination. Furthermore, the semi-parameter is constrained to the initial value according to

$$p_L = p_U = p_i$$

If the park orbit is circular, the software enforces the following two constraints.

$$f = 0 \qquad g = 0$$

Otherwise, for an elliptical park orbit, the single constraint $\sqrt{f^2 + g^2} = e$ is enforced, where *e* is the user-defined park orbit eccentricity.

For constraint option 2, both lower and upper bounds for the other modified equinoctial elements are set equal to the initial elements.

$$f_L = f_U = f_i \qquad g_L = g_U = g_i$$
$$h_L = h_U = h_i \qquad k_L = k_U = k_i$$
$$L_L = L_U = L_i$$

Option 3 is identical to option 2 with the initial true longitude bounded according to

$$-180^\circ \le L_i \le +180^\circ$$

In optimal control terminology, these constraints or boundary conditions are called *point functions*.

Performance index – maximize final spacecraft mass

The objective function or performance index *J* for this simulation is the mass of the spacecraft at burnout of the interplanetary injection stage. This is simply $J = m_f$.

The value of the `maxmin` indicator in the *Sparse Optimization Suite* tells the software whether the user is minimizing or maximizing the performance index. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass. The spacecraft mass at the initial time is constrained to the user-defined initial value.

Please note that the `hyper_sos` software will not be able to optimize the propulsive maneuver using the fixed attitude steering option and a highly constrained initial park orbit. For this situation, the software will display a warning and internally set `maxmin` to zero (no optimization).

Path constraint – unit thrust vector scalar magnitude

For the *variable steering* program option, the scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as

$$\left|\mathbf{u}_T\right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

Point functions – final "scaled" asymptote unit vector

At the final time, the solution should satisfy the following "energy scaled" unit asymptote *targeted minus predicted* vector equality constraint

$$\left(C_3\,\hat{\mathbf{s}}\right)_T - \left(C_3\,\hat{\mathbf{s}}\right)_P = 0$$

where the user-defined "target" departure asymptote unit vector is given by

$$\hat{\mathbf{s}}_T = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

In this expression, α_∞ is the user-defined right ascension (RLA) and δ_∞ is the user-defined declination (DLA) of the hyperbolic departure orbit.

The predicted "twice specific" (per unit mass) orbital energy (C3) is determined from the final position **r** and velocity **v** vectors according to

$$C_3 = |\mathbf{v}|^2 - \frac{2\mu}{|\mathbf{r}|}$$

The predicted asymptote unit vector at any trajectory time can be computed from

$$\hat{\mathbf{s}}_P = \frac{1}{1 + C_3 \dfrac{h^2}{\mu^2}} \left\{ \left(\frac{\sqrt{C_3}}{\mu} \right) \mathbf{h} \times \mathbf{e} - \mathbf{e} \right\} = \frac{1}{1 + C_3 \dfrac{p}{\mu}} \left\{ \left(\frac{\sqrt{C_3}}{\mu} \right) \mathbf{h} \times \mathbf{e} - \mathbf{e} \right\}$$

where **h** and **e** are the predicted final angular momentum and orbital eccentricity vectors, respectively.

These orientation vectors are computed using the following two equations.

$$\mathbf{h} = \mathbf{r} \times \mathbf{v}$$

$$\mathbf{e} = \frac{\mathbf{v} \times \mathbf{h}}{\mu} - \frac{\mathbf{r}}{r} = \frac{1}{\mu} \left[\left(v^2 - \frac{\mu}{r} \right) \mathbf{r} - (\mathbf{r} \bullet \mathbf{v}) \mathbf{v} \right]$$

Bounds on the dynamic variables

The following lower and upper bounds are applied to the spacecraft mass and the modified equinoctial dynamic variables *during* the propulsive maneuver.

$$0.05 m_{sc_i} \le m_{sc} \le 1.05 m_{sc_i} \qquad p \ge 0.8 p_i$$

$$-1 \le h \le +1 \qquad -1 \le k \le +1$$

where m_{sc_i} is the initial spacecraft mass. The elements *f* and *g* are unconstrained.

For the variable steering option, the radial, tangential, and normal components of the unit thrust vector are constrained as follows

$$-1.1 \le u_r \le +1.1 \qquad -1.1 \le u_t \le +1.1 \qquad -1.1 \le u_n \le +1.1$$

For the inertially fixed attitude steering option, the right ascension and declination angles are bounded as

$$-180^\circ \le \alpha \le +180^\circ \qquad -90^\circ \le \delta \le +90^\circ$$

Technical discussion

In this computer program, the orbital motion of the spacecraft is modeled in the modified equinoctial orbital elements coordinate system. Please consult *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for a definition of these orbital elements along with the equations of motion in this system. *Appendix G – Aerospace Trajectory Coordinates and Time Systems* also provides useful technical details about coordinate transformations.

Propulsive thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)} \hat{\mathbf{u}}_T(t)$$

where T is the constant thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_r} & u_{T_t} & u_{T_n} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system.

For the variable steering option, the components of this unit vector are control variables.

The propellant mass flow rate is determined from

$$\dot{m} = \frac{dm}{dt} = \frac{T}{g\, I_{sp}}$$

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g\, I_{sp}$ is also called the *exhaust velocity*. The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}\, t$ where m_{sc_i} is the initial mass of the spacecraft.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

Finally, the pitch and yaw angles can be determined from the components of the unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the "local horizontal" and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four-quadrant operation.

For either steering option, the software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems. The following section summarizes the inertial-to/from-modified equinoctial coordinate transformations and the calculation of the inertial unit thrust vector in terms of right ascension and declination angles.

In the `hyper_sos` computer program, the components of the inertial unit thrust vector are defined in terms of the right ascension α and the declination angle δ as follows:

$$u_{T_{ECI_x}} = \cos\alpha\cos\delta \qquad u_{T_{ECI_y}} = \sin\alpha\cos\delta \qquad u_{T_{ECI_z}} = \sin\delta$$

The right ascension and declination angles can be determined from the components of the ECI unit thrust vector according to

$$\alpha = \tan^{-1}\left(u_{T_{ECI_y}}, u_{T_{ECI_x}}\right) \qquad \delta = \sin^{-1}\left(u_{T_{ECI_z}}\right)$$

The right ascension and declination angles are treated as problem parameters in the fixed inertial attitude steering option.

Finite-burn solutions

This section describes how to use `hyper_sos` to optimize the finite-burn trajectories for each of the hyperbolic injection opportunities described in Appendix H.

For an initial spacecraft mass of 4000 kilograms, a specific impulse of 450 seconds, and a delta-v of 3641.245 meters/second, the propellant mass required for the impulsive maneuver is given by

$$m_p = m_i\left(1 - e^{\frac{-\Delta V}{V_{ex}}}\right) = 4000 \bullet \left(1 - e^{\frac{-3641.245}{9.80665 \cdot 450}}\right) = 2247.27 \text{ kilograms}$$

For a thrust level of 19840 newtons, the calculation for thrust duration is

$$t_d = \frac{g\, I_{sp}\, m_p}{F} = \frac{9.80665 \bullet 450 \bullet 2247.27}{19480} = 499.58 \text{ seconds}$$

With this answer for the thrust duration, we can create the following (partial) input data file for `hyper_sos`. Notice that we use a thrust duration initial guess of 550 seconds and the park orbit RAAN (176 degrees) of the first opportunity. We also constrain only the semimajor axis, eccentricity and orbital inclination of the initial Earth park orbit.

```
**************************************************************************
** Optimal Finite-Burn Earth Orbit-to-Interplanetary Injection
** single phase, continuous-thrust maneuver with user-defined final coast
** variable and fixed inertial attitude steering
** input file ==> hyper2.in - February 14, 2011
**************************************************************************

type of steering
1 = fixed inertial attitude
2 = variable attitude
---------------------
2

initial guess for right ascension (degrees)
0.0d0

initial guess for declination (degrees)
0.0d0

initial spacecraft mass (kilograms)
4000.0

thrust magnitude (newtons)
19840.0

specific impulse (seconds)
450.0

initial guess for thrust duration (seconds)
550.0
```

```
lower bound for thrust duration (seconds)
1.0
upper bound for thrust duration (seconds)
1000.0

*****************
* INITIAL ORBIT *
*****************

semimajor axis (kilometers)
6563.34

orbital eccentricity (non-dimensional)
0.0

orbital inclination (degrees)
28.5

argument of perigee (degrees)
0.0

right ascension of the ascending node (degrees)
176

true anomaly (degrees)
0.0

*******************************
initial orbit constraint options
*******************************
1 = constrain semimajor axis, eccentricity and inclination
2 = constrain all initial orbital elements
3 = option 2 with unconstrained true longitude
-----------------------------------------------
1

********************
* LAUNCH HYPERBOLA *
********************

specific orbital energy (C3; [km/sec]**2)
9.28

right ascension of outgoing asymptote (RLA; degrees)
352.59

declination of outgoing asymptote (DLA; degrees)
2.27

final coast duration (seconds)
100.0

************************
* type of gravity model *
------------------------
1 = spherical Earth
2 = oblate Earth
----------------
2

************************
* initial guess options *
************************
 1 = numerical integration
 2 = binary restart file
------------------------
1
```

Here is the `hyper_sos` solution for the first injection opportunity.

```
 program hyper_sos
 ==================

 input file ==> hyper2.in
```

numerical integration initial guess

variable attitude steering

oblate earth gravity model

```
------------------------
beginning of finite burn
------------------------

     sma (km)            eccentricity       inclination (deg)       argper (deg)
 0.656334000000D+04    0.127163366083D-16    0.285000000000D+02    0.000000000000D+00

    raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
 0.176807800508D+03    0.578903358972D+01    0.578903358972D+01    0.881956335064D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.655213255829D+04    -.217269772647D+03    0.315887226893D+03    0.656334000000D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.405405291239D+00    -.684692084631D+01    0.369954899383D+01    0.779303158492D+01

------------------
end of finite burn
------------------

     sma (km)            eccentricity       inclination (deg)       argper (deg)
 -.429123585603D+05    0.115435598318D+01    0.284800431601D+02    0.251836707509D+02

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.176786768745D+03    0.208841694470D+02    0.460678401979D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.499940493858D+04    -.407207922203D+04    0.235767389573D+04    0.686545738673D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.593950144357D+01    -.842217918403D+01    0.438127782797D+01    0.111984987839D+02

final mass               1741.10030901186      kilograms

propellant mass          2258.89969098814      kilograms

thrust duration          502.444929161942      seconds
                         8.37408215269904      minutes

delta-v                  3670.62646757410      meters/second

------------------
end of final coast
------------------

     sma (km)            eccentricity       inclination (deg)       argper (deg)
 -.429526337823D+05    0.115421436692D+01    0.284776394025D+02    0.251931783309D+02

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.176780639937D+03    0.297298409658D+02    0.549230192967D+02

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.437687317644D+04    -.488843426872D+04    0.278088318892D+04    0.712650824125D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.648875162967D+01    -.789882809086D+01    0.408029172582D+01    0.110065509352D+02

------------------
outgoing hyperbola
------------------

right ascension          352.590000000000      degrees

declination              2.27000000000005      degrees

orbital energy           9.27999999999990      (km/sec)**2

final coast duration     100.000000000000      seconds
```

```
                        1.66666666666667      minutes

 ----------------------------
 verification of SOS solution
 ----------------------------

 right ascension         352.590000136648      degrees

 declination             2.26999992314201      degrees

 orbital energy          9.27999983738958      (km/sec)**2


 final mass              1741.10030901414      kilograms

 propellant mass         2258.89969098586      kilograms

 delta-v                 3670.62604328145      meters/second
```

Here is the `hyper_sos` solution for the second opportunity. The only change to the input data file involved setting the RAAN of the park orbit to 348 degrees.

```
 program hyper_sos
 ==================

 input file ==> hyper2.in

 numerical integration initial guess

 variable attitude steering

 oblate earth gravity model

 ------------------------
 beginning of finite burn
 ------------------------

      sma (km)              eccentricity       inclination (deg)       argper (deg)
  0.656334000000D+04    0.337784797333D-15    0.285000000000D+02    0.000000000000D+00

     raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
  0.348438209294D+03    0.195324084717D+03    0.195324084717D+03    0.881956335064D+02

      rx (km)                ry (km)              rz (km)              rmag (km)
  -.650706632841D+04    -.224735623270D+03    -.827655515878D+03    0.656334000000D+04

      vx (kps)               vy (kps)             vz (kps)             vmag (kps)
  0.693905143549D+00    -.688391171243D+01    -.358630680098D+01    0.779303158492D+01

 ------------------
 end of finite burn
 ------------------

      sma (km)              eccentricity       inclination (deg)       argper (deg)
  -.429180375328D+05    0.115433556073D+01    0.284786048967D+02    0.214713200321D+03

     raan (deg)         true anomaly (deg)       arglat (deg)
  0.348406880900D+03    0.208981941128D+02    0.235611394434D+03

      rx (km)                ry (km)              rz (km)              rmag (km)
  -.479953648441D+04    -.409934688621D+04    -.270163893604D+04    0.686578825041D+04

      vx (kps)               vy (kps)             vz (kps)             vmag (kps)
  0.626270382532D+01    -.846323354961D+01    -.381467808151D+01    0.111981940591D+02

  final mass              1741.03235436989      kilograms
 propellant mass          2258.96764563011      kilograms

  thrust duration         502.460044249510      seconds
                          8.37433407082516      minutes

  delta-v                 3670.79881229368      meters/second
```

```
------------------
end of final coast
------------------

     sma (km)             eccentricity       inclination (deg)       argper (deg)
 -.429526337823D+05    0.115421378015D+01    0.284764855902D+02    0.214723293631D+03

    raan (deg)          true anomaly (deg)       arglat (deg)
 0.348399158345D+03    0.297436442833D+02    0.244466937914D+03

     rx (km)                ry (km)              rz (km)              rmag (km)
 -.414594745792D+04    -.491964777121D+04    -.306625412665D+04    0.712697192963D+04

     vx (kps)               vy (kps)             vz (kps)             vmag (kps)
 0.678656978359D+01    -.793668058918D+01    -.347684588160D+01    0.110062203088D+02

------------------
outgoing hyperbola
------------------

right ascension          352.590000000000      degrees

declination              2.27000000000001      degrees

orbital energy           9.27999999999994      (km/sec)**2


final coast duration     100.000000000000      seconds
                         1.66666666666667      minutes

----------------------------------------
verification of optimal control solution
----------------------------------------

right ascension          352.590000184595      degrees

declination              2.27000008928881      degrees

orbital energy           9.27999983772533      (km/sec)**2

final mass               1741.03235436945      kilograms

propellant mass          2258.96764563055      kilograms

delta-v                  3670.79828447177      meters/second
```

The finite-burn solution requires a delta-v which is about 29 meters per second larger than the impulsive solution.

Fixed thrust duration program example

The `hyper_sos` software can also be used to solve the Earth orbit-to-interplanetary injection problem using a fixed thrust duration maneuver. This section illustrates how to modify the data file for the variable thrust duration example given previously in this document to solve the same problem using a fixed thrust duration maneuver.

Since the thrust duration is fixed, the propellant required is determined by the burn time and the software is essentially solving a two-point boundary value problem (TPBVP). An estimate for the fixed thrust duration time can be determined a priori from the optimal variable thrust time solution.

This simulation capability can be applied to missions that perform this type of orbital maneuver using solid propellant upper or “kick” stages. These missions typically customize the performance of the stage by offloading propellant or perhaps by modifying the geometry of the solid propellant grain. The stage usually includes extra propellant to account for dispersions in propulsive performance due to thrust,

specific impulse or other variations. The statistical stage performance can be determined from a Monte Carlo dispersion analysis.

In this example, the thrust duration is fixed at 520 seconds which is slightly longer than the optimal variable thrust duration answer of 497.258 seconds. These next several lines are the changes to the input data file used for the variable thrust example. The first part of the data fixes the lower and upper bounds for the thrust duration. The second part of the data invokes the compressed Hermite-Simpson solution collocation method.

```
lower bound for thrust duration (seconds)
520.0

upper bound for thrust duration (seconds)
520.0

discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
3
```

The following is the numerical output for this fixed thrust duration example.

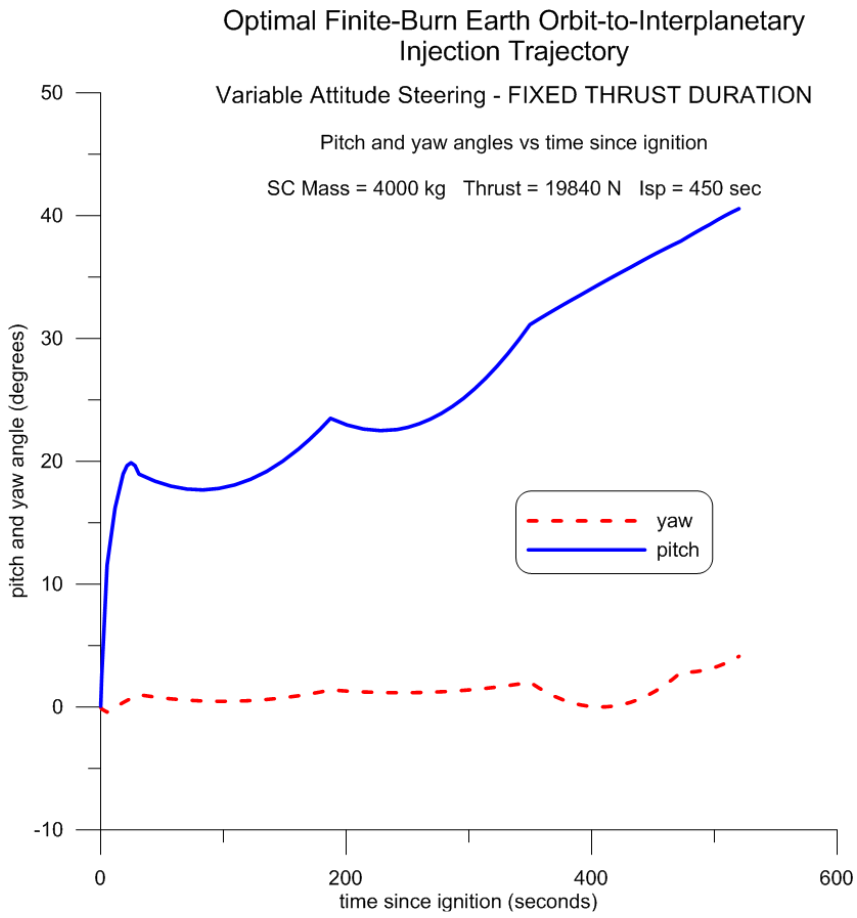
```
 program hyper_sos
 =================

 input file ==> hyper1_ftd.in

 numerical integration initial guess

 variable attitude steering

 oblate earth gravity model

 ------------------------
 beginning of finite burn
 ------------------------

      sma (km)             eccentricity       inclination (deg)       argper (deg)
  0.656334000000D+04    0.150000000002D-01    0.285000000000D+02    0.161713758267D+03

     raan (deg)         true anomaly (deg)       arglat (deg)          period (min)
  0.287782013241D+01    0.287316640370D+02    0.190445422304D+03    0.881956335064D+02

      rx (km)                ry (km)              rz (km)              rmag (km)
  -.630950164991D+04    -.135039878807D+04    -.560286245754D+03    0.647667419547D+04

      vx (kps)               vy (kps)             vz (kps)             vmag (kps)
  0.171769210648D+01    -.675574572790D+01    -.371026868171D+01    0.789662334185D+01

 ------------------
 end of finite burn
 ------------------

      sma (km)             eccentricity       inclination (deg)       argper (deg)
  -.453147598855D+05    0.114087135765D+01    0.281389717352D+02    0.194505540587D+03

     raan (deg)         true anomaly (deg)       arglat (deg)
  0.231005534166D+01    0.372841472855D+02    0.231789687872D+03
      rx (km)                ry (km)              rz (km)              rmag (km)
  -.422743762281D+04    -.513812279108D+04    -.265462981190D+04    0.716369975007D+04

      vx (kps)               vy (kps)             vz (kps)             vmag (kps)
  0.611281335370D+01    -.796576227098D+01    -.438859972999D+01    0.109580866599D+02

 final mass              1662.17594970650      kilograms

 propellant mass         2337.82405029350      kilograms
```

```
thrust duration          520.000000000000      seconds
                         8.66666666666667      minutes

delta-v                  3875.34421231626      meters/second

------------------
end of final coast
------------------

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.453544448786D+05    0.114075081311D+01    0.281369896983D+02    0.194512674908D+03

    raan (deg)         true anomaly (deg)       arglat (deg)
 0.230388918141D+01    0.450918704277D+02    0.239604545335D+03

     rx (km)                 ry (km)              rz (km)              rmag (km)
 -.359545119031D+04    -.590696682842D+04    -.307908064562D+04    0.756969378483D+04

     vx (kps)                vy (kps)             vz (kps)             vmag (kps)
 0.650555427099D+01    -.741354952101D+01    -.410127099902D+01    0.106819182117D+02

------------------
outgoing hyperbola
------------------

right ascension          349.680040000000      degrees

declination             -6.66625300000002      degrees

orbital energy           8.78856400000011      (km/sec)**2

final coast duration     100.000000000000      seconds
                         1.66666666666667      minutes
```

The following is the verification of this solution.

```
---------------------------------------
verification of optimal control solution
---------------------------------------

right ascension          349.680038049253      degrees

declination             -6.66625366350336      degrees

orbital energy           8.78856398999962      (km/sec)**2


final mass               1662.17594976651      kilograms

propellant mass          2337.82405023349      kilograms

delta-v                  3875.34351982112      meters/second
```

This plot illustrates the behavior of the pitch and yaw control angles for this example. It appears that the "boundary value" solution modifies both the inplane and out-of-plane steering angles in order to manage the excess energy.



Here is the program output for the same fixed thrust duration example with fixed inertial attitude steering. This type of solution might be used for a "point and shoot" upper stage or kick motor.

```
program hyper_sos
=================

input file ==> hyper1_ftd.in

numerical integration initial guess

fixed inertial attitude steering

oblate earth gravity model

------------------------
beginning of finite burn
------------------------

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 0.656334000017D+04    0.150000008821D-01    0.285000000211D+02    0.811674903919D+01

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.191587461675D+01    0.180376794339D+03    0.188493543379D+03    0.881956335099D+02

     rx (km)                ry (km)               rz (km)               rmag (km)
 -.655613307360D+04    -.108448816061D+04    -.469492000668D+03    0.666178791226D+04

     vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 0.135708221832D+01    -.663092375637D+01    -.362291920429D+01    0.767700238184D+01

------------------
end of finite burn
------------------
```

```
      sma (km)              eccentricity       inclination (deg)       argper (deg)
  -.453142457306D+05    0.114486576255D+01    0.286230517643D+02    0.195093043483D+03

     raan (deg)         true anomaly (deg)       arglat (deg)
  0.205290920790D+01    0.331044392275D+02    0.228197482710D+03

      rx (km)                ry (km)               rz (km)              rmag (km)
  -.461919823208D+04    -.487151722229D+04    -.256657032633D+04    0.718720777453D+04

      vx (kps)               vy (kps)              vz (kps)             vmag (kps)
  0.584533514713D+01    -.807154659999D+01    -.451641098329D+01    0.109414704527D+02

 final mass              1662.17594976651      kilograms

 propellant mass         2337.82405023349      kilograms

 thrust duration         520.000000000000      seconds
                         8.66666666666667      minutes

 delta-v                 3875.34615240030      meters/second

 inertial steering angles

 right ascension        -79.6000467698613      degrees

 declination            -28.6839033728582      degrees

 ------------------
 end of final coast
 ------------------

      sma (km)              eccentricity       inclination (deg)       argper (deg)
  -.453544448781D+05    0.114474018459D+01    0.286210014275D+02    0.195100029212D+03

     raan (deg)         true anomaly (deg)       arglat (deg)
  0.204739189689D+01    0.410195855400D+02    0.236119614752D+03

      rx (km)                ry (km)               rz (km)              rmag (km)
  -.401200354971D+04    -.565240333034D+04    -.300429349712D+04    0.755457578614D+04

      vx (kps)               vy (kps)              vz (kps)             vmag (kps)
  0.627767300123D+01    -.754630061265D+01    -.423772325580D+01    0.106917786004D+02

------------------
 outgoing hyperbola
 ------------------

 right ascension         349.680040000129      degrees

 declination            -6.66625300115772      degrees

 orbital energy          8.78856400010413      (km/sec)**2

final coast duration    100.000000000000      seconds
                         1.66666666666667      minutes

----------------------------------------
 verification of optimal control solution
 ----------------------------------------

 right ascension         349.680041062561      degrees

 declination            -6.66625242714642      degrees

 orbital energy          8.78856316301803      (km/sec)**2


 final mass              1662.17594976651      kilograms

 propellant mass         2337.82405023349      kilograms

 delta-v                 3875.34351982113      meters/second
```

Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and CSV data file produced by the `hyper_sos` software.

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (min) = orbital period in minutes. If the orbit is hyperbolic, the period is undefined and a value of 0 is displayed.

rx (km) = x-component of the eci position vector in kilometers

ry (km) = y-component of the eci position vector in kilometers

rz (km) = z-component of the eci position vector in kilometers

rmag (km) = geocentric position magnitude in kilometers

vx (kps) = x-component of the eci velocity vector in kilometers/second

vy (kps) = y-component of the eci velocity vector in kilometers/second

vz (kps) = z-component of the eci velocity vector in kilometers/second

vmag (kps) = velocity vector scalar magnitude in kilometers/seconds

The user-defined comma-separated-variable (CSV) disk file is created by the software and contains the following information.

time (sec) = time since ignition in seconds

time (min) = time since ignition in minutes

semimajor axis (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

arg of perigee (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (min) = orbital period in minutes

mass (kg) = spacecraft mass in kilograms

thracc (m/s2)** = thrust acceleration in meters per second squared

yaw (deg) = thrust vector yaw angle in degrees

pitch (deg) = thrust vector pitch angle in degrees

rasc (deg) = thrust vector right ascension angle in degrees

decl (deg) = thrust vector declination angle in degrees

perigee alt (km) = perigee altitude in kilometers

apogee alt (km) = apogee altitude in kilometers

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

ut-eci-x = x-component of the eci unit thrust vector

ut-eci-y = y-component of the eci unit thrust vector

ut-eci-z = z-component of the eci unit thrust vector

semi-parameter (km) = orbital semiparameter in kilometers

f equinoctial element = ecc * cos(argper + raan)

g equinoctial element = ecc * sin(argper + raan)

h equinoctial element = tan(i/2) * cos(raan)

k equinoctial element = tan(i/2) * sin(raan)

true longitude (deg) = orbital true longitude in degrees

rx (km) = x-component of eci position vector in kilometers

ry (km) = y-component of eci position vector in kilometers

rz (km) = z-component of eci position vector in kilometers

rmag (km) = geocentric radius magnitude in kilometers

vx (kps) = x-component of eci velocity vector in kilometers per second

vy (kps) = y-component of eci velocity vector in kilometers per second

vz (kps) = z-component of eci velocity vector in kilometers per second

vmag (kps) = scalar velocity vector in kilometers per second

fpa (deg) = flight path angle in degrees

c33 (km/sec)2** = twice specific orbital energy

shat(1) = x-component of c3-scaled unit asymptote vector

shat(2) = y-component of c3-scaled unit asymptote vector

shat(3) = z-component of c3-scaled unit asymptote vector

deltav (mps) = accumulative delta-v in meters per second

"Optimal Finite-Thrust Spacecraft Trajectories Using Direct Transcription and Nonlinear Programming", Paul J. Enright, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1991.

"Improved Collocation Methods with Application to Direct Trajectory Optimization", Albert L. Herman, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1995.

"Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

CMATO 17 – Low-thrust Earth Escape Trajectory Optimization

This *CMATO* application is a Fortran computer program named `escape_sos` that uses the *Sparse Optimization Suite (SOS)* distributed by Applied Mathematical Analysis to solve the low-thrust finite-burn, Earth escape trajectory optimization problem. The software models the trajectory as a single, finite-burn propulsive maneuver. This computer program attempts to maximize the final spacecraft mass. Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the propellant mass required to perform this orbit transfer.

The important features of this scientific simulation are as follows:

- single, continuous thrust orbital maneuver
- variable attitude steering
- constant propulsive thrust magnitude
- modified equinoctial equations of motion with oblate Earth gravity model
- user-defined final orbital energy and orbital inclination
- numerical verification of the optimal control solution

The *Sparse Optimization Suite* is a *direct transcription method* that can be used to solve a variety of trajectory optimization problems using the following combination of numerical methods:

- collocation and *implicit* integration
- adaptive mesh refinement
- sparse nonlinear programming

Additional information about the mathematical techniques and numerical methods used in the *Sparse Optimization Suite* can be found in the book, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* by John. T. Betts, SIAM, 2010.

The `escape_sos` software consists of Fortran routines that perform the following tasks:

- set algorithm control parameters and call the transcription/optimal control subroutine
- define the problem structure and perform initialization related to scaling, lower and upper bounds, initial conditions, etc.
- compute the *right-hand-side* differential equations
- evaluate any point and path constraints
- display the optimal solution results and create an output file

The *Sparse Optimization Suite* will use this information to *automatically* transcribe the user's optimal control problem and perform the optimization using a sparse nonlinear programming (NLP) method.

The `escape_sos` software allows the user to select the type of initial guess, collocation method, and other important algorithm control parameters.

Input file format and contents

The `escape_sos` software is "data-driven" by a user-created text file. The following is a typical input file used by this computer program. In the following discussion the actual input file contents are in courier font and all explanations are in times font. This example attempts to optimize the maneuver required to perform a low-thrust escape from a 1000 kilometer circular Earth orbit.

Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct units and when appropriate, the valid range of the input. ASCII text input is not case sensitive but must be spelled correctly.

The first six lines of any input file are reserved for user comments. These lines are ignored by the software. However, the input file must begin with six and only six initial text lines.

```
****************************************
** earth escape trajectory optimization
** single phase, continous-thrust maneuver
** program escape_sos
** leo2esc.in - April 25, 2012
****************************************
```

The first three inputs provide the initial spacecraft mass, thrust magnitude (assumed constant) and the specific impulse (also assumed constant) of the propulsion system.

```
initial spacecraft mass (kilograms)
1000.0

thrust magnitude (newtons)
5.0

specific impulse (seconds)
3300.0
```

The next two inputs define the circular orbit altitude and inclination of the initial Earth orbit.

```
*****************
* INITIAL ORBIT *
*****************

altitude (kilometers)
1000.0

orbital inclination (degrees)
28.5
```

The next two inputs specify the orbital inclination and specific orbital energy of the final escape trajectory.

```
***************
* FINAL ORBIT *
***************

orbital inclination (degrees)
10.0

specific orbital energy (km/sec)**2
0.0
```

This next input defines the method used to estimate the transfer time.

```
****************************************
* type of initial guess for transfer time *
----------------------------------------
 1 = numerical integration
 2 = user-defined
-----------------
1
```

For option 2 above, the next input is the user's initial guess for the transfer time.

```
******************************************************
* user-defined initial guess for transfer time (hours) *
******************************************************
480
```

The next integer input defines the type of initial guess to use for the trajectory.

```
************************
* initial guess options *
************************
 1 = numerical integration
 2 = binary data file
---------------------
1
```

This input defines the name of a restart or initial guess binary data file.

```
name of binary initial guess data file
leo2esc.rsbin
```

The following input can be used to create or update an initial guess binary file. The creation or update process uses the filename defined above. For initial guess option 1, the software will create a binary restart file. For initial guess option 2, an input of `yes` to this item will update the binary file used to initialize the simulation.

```
*****************************
* binary restart file option *
*****************************

create/update binary data file (yes or no)
no
```

This next input specifies the type of solution data file to create.

```
*******************************************
* type of comma-delimited solution data file *
*******************************************
 1 = SOS-defined nodes
 2 = user-defined nodes
 3 = user-defined step size
---------------------------
1
```

For options 2 or 3, this input defines either the number of data points or the time step size of the data output in the solution file.

```
number of user-defined nodes or print step size in solution data file
50
```

The name of the comma-separated-variable solution data file is defined in this next line.

```
name of solution output file
leo2esc.csv
```

The next series of program inputs are algorithm control options and parameters for the *SOS* software. The first input is an integer that specifies the type of collocation method to use during the solution process. For most simulations, the trapezoidal method is recommended.

```
discretization/collocation method
---------------------------------
 1 = trapezoidal
 2 = separated Hermite-Simpson
 3 = compressed Hermite-Simpson
-------------------------------
1
```

The next input defines the relative error in the objective function.

```
*******************************
* algorithm control parameters *
*******************************

relative error in the objective function (performance index)
1.0d-5
```

The next input defines the relative error in the solution of the differential equations.

```
relative error in the solution of the differential equations
1.0d-7
```

The next input is an integer that defines the maximum number of mesh refinement iterations.

```
maximum number of mesh refinement iterations
20
```

The next input is an integer that defines the maximum number of function evaluations.

```
maximum number of function evaluations
50000
```

The next input is an integer that defines the maximum number of algorithm iterations.

```
maximum number of algorithm iterations
10000
```

The level of output from the *Sparse Optimization Suite* NLP algorithm is controlled with the following integer input.

```
*************************
sparse NLP iteration output
---------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
2
```

The level of output from the *Sparse Optimization Suite* optimal control algorithm is controlled with the following integer input. Please note that option 4 will create lots of information.

```
**********************
optimal control output
----------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
-----------------
1
```

The level of output from the *Sparse Optimization Suite* differential equations algorithm is controlled with the following integer input. Please note that option 5 will create lots of information.

```
***************************
differential equation output
----------------------------
 1 = none
 2 = terse
 3 = standard
 4 = interpretive
 5 = diagnostic
---------------
1
```

The level of output can be further controlled by the user with this final text input. This program option sets the value of the SOCOUT character variable described in the *Sparse Optimization Suite* user's manual. To ignore this special output control, input the simple character string `no`.

```
******************
user-defined output
-------------------
input no to ignore
------------------
a0b0c0d0e0f0g0h0i0j2k0l0m0n0o0p0q0r0
```

The last series of inputs allow the reading and writing of configuration input files. The user should create a configuration file before attempting to read one. These configuration files are simple text files which can be edited external to the `escape_sos` software. Please read *Appendix J – Sparse Optimization Suite Configuration File* for more information about this file.

```
************************************
* optimal control configuration options
************************************

read an optimal control configuration file (yes or no)
no

name of optimal control configuration file
leo2esc_config.txt

create an optimal control configuration file (yes or no)
no

name of optimal control configuration file
leo2esc_config1.txt
```

Optimal control solution

The following is the `escape_sos` solution for this example. This simulation used the numerical integration method for estimating the transfer time, and the tangential steering method to create the initial guess for the transfer trajectory. The output includes the orbital characteristics at the beginning and end of the propulsive maneuver.

```
 program escape_sos
 ------------------

 input file ==> leo2esc.in

 numerical integration initial guess

------------------------
 beginning of finite burn
 ------------------------

      sma (km)              eccentricity       inclination (deg)       argper (deg)
  0.737814000000D+04    0.333066907388D-15    0.285000000000D+02    0.000000000000D+00

     raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
  0.000000000000D+00    0.000000000000D+00    0.000000000000D+00    0.105118713217D+03

      rx (km)                 ry (km)              rz (km)              rmag (km)
  0.737814000000D+04    0.000000000000D+00    0.000000000000D+00    0.737814000000D+04

      vx (kps)               vy (kps)              vz (kps)              vmag (kps)
  0.000000000000D+00    0.645942676649D+01    0.350718257926D+01    0.735013767191D+01

 ------------------
 end of finite burn
 ------------------

      sma (km)              eccentricity       inclination (deg)       argper (deg)
  -.216133734515D+16    0.100000000006D+01    0.100000000029D+02    0.310802842395D+03

     raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
  0.346516988187D+03    0.823718561078D+02    0.331746985029D+02    0.000000000000D+00

      rx (km)                 ry (km)              rz (km)              rmag (km)
  0.224184866614D+06    0.784697741316D+05    0.226716572171D+05    0.238600846419D+06

      vx (kps)               vy (kps)              vz (kps)              vmag (kps)
  0.663338212631D+00    0.167401922892D+01    0.314310552364D+00    0.182788103729D+01
```

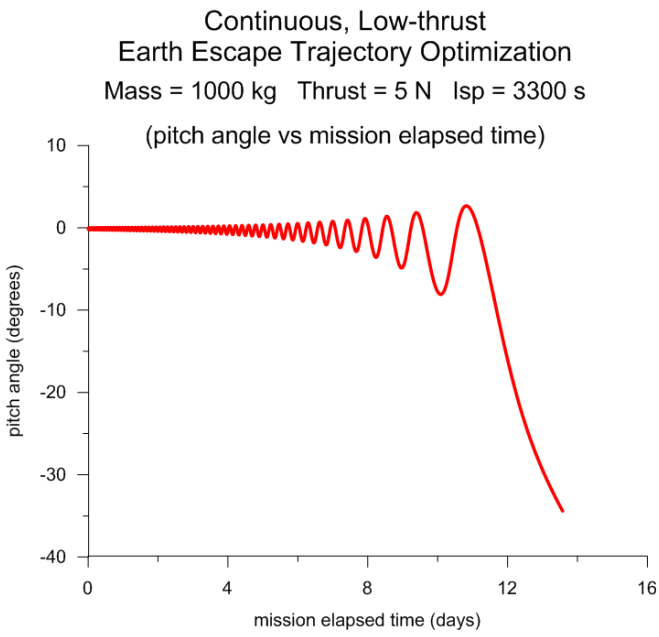
The following program output is the final spacecraft mass, the propellant mass consumed, the actual thrust duration for the maneuver, the accumulated delta-v and the final specific orbital energy.

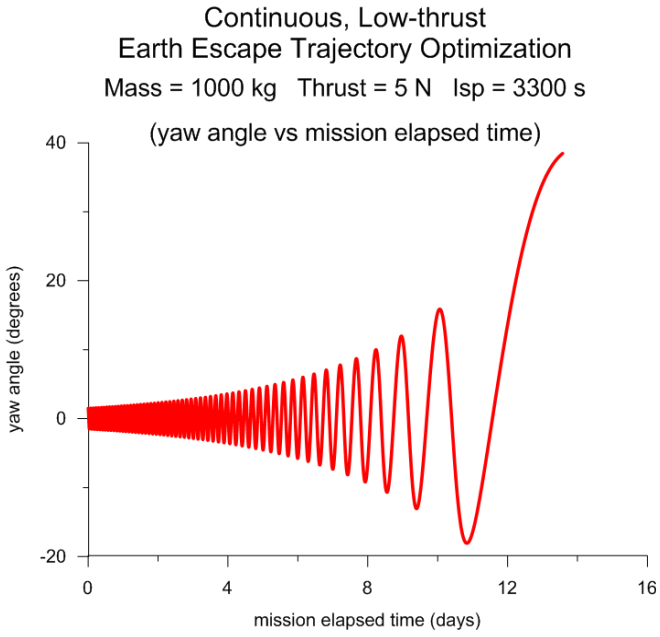
```
 final mass             818.781528480986      kilograms

 propellant mass        181.218471519014      kilograms

 thrust duration        325.810122682355      hours
                        13.5754217784315      days

 delta-v                6470.38206545964      meters/second

 specific energy       1.844226993341636E-010 km**2/sec**2
```

The delta-v is estimated using a cubic spline method that integrates the thrust acceleration at each collocation grid point of the solution. Notice that the final orbital eccentricity is equal to one which confirms that a parabolic escape orbit has been created.

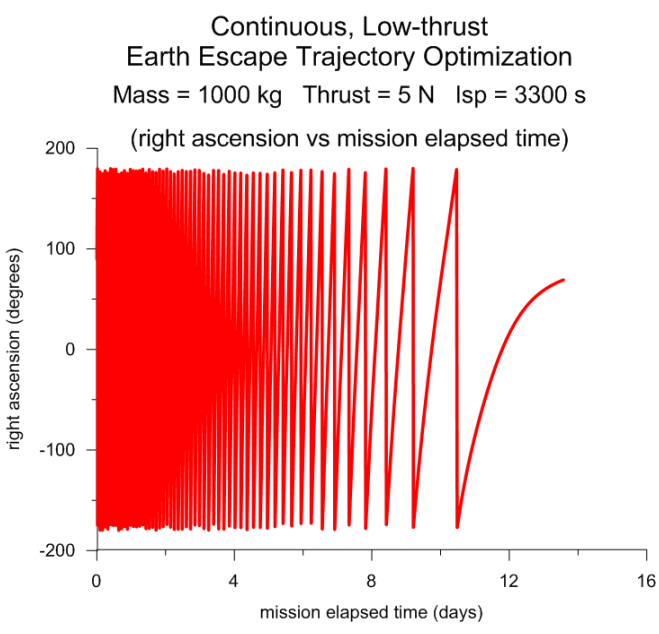
The `escape_sos` computer program will create a comma-separated-variable (csv) data file containing the optimal control solution. This data file can be used to create graphic displays of important trajectory characteristics. Additional information about the contents of this file can be in the *Contents of the simulation summary and csv files* section later in this document.

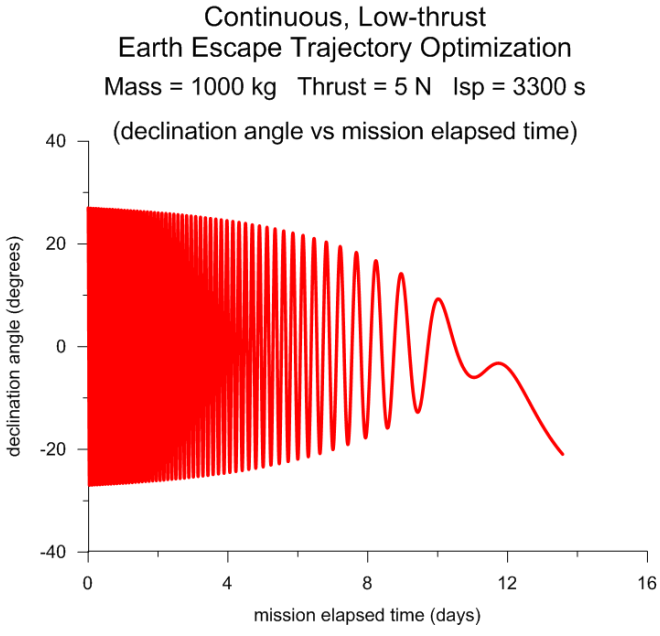
The following two plots illustrate the evolution of the pitch and yaw steering angles during this finite-burn maneuver.



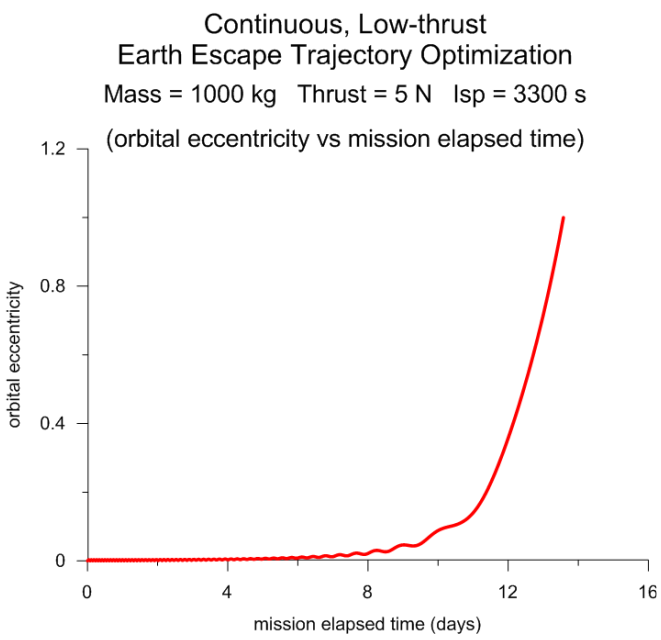


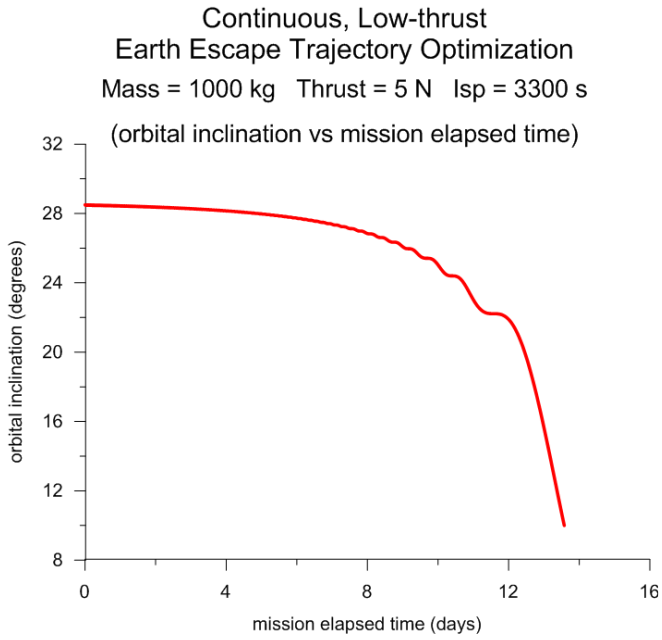
The next two plots illustrate the evolution of the inertial right ascension and declination angles of the thrust vector.



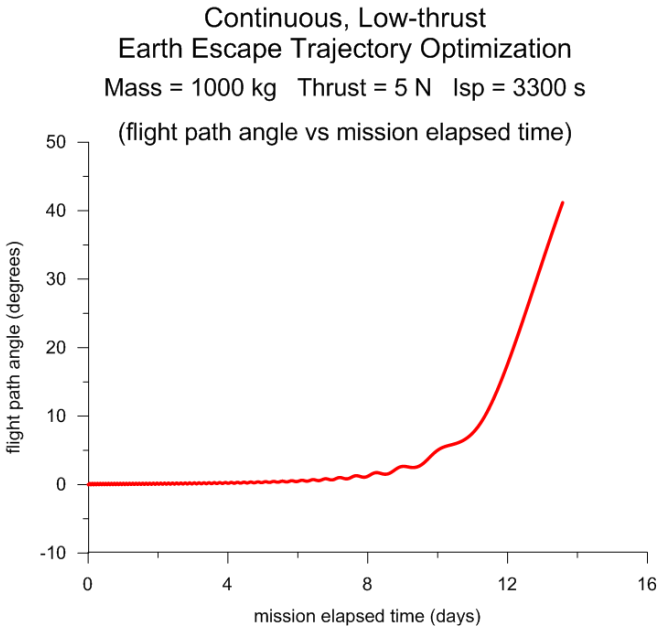


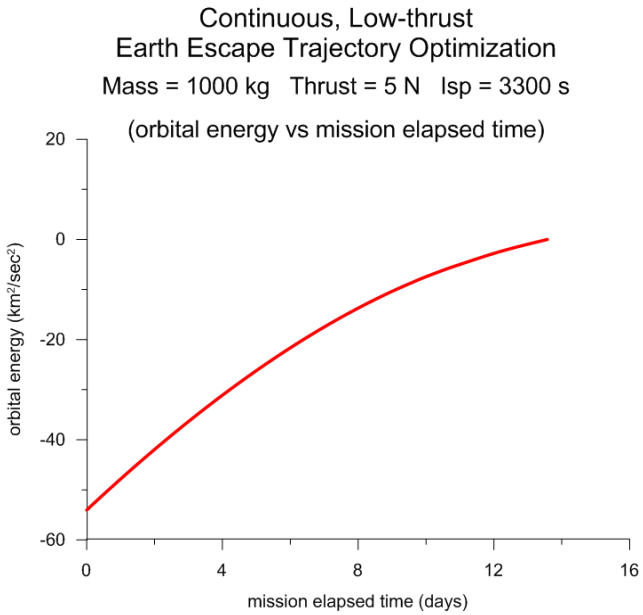
The next two plots illustrate the behavior of the orbital eccentricity and inclination during the low-thrust escape trajectory. As expected the final orbital eccentricity is one.



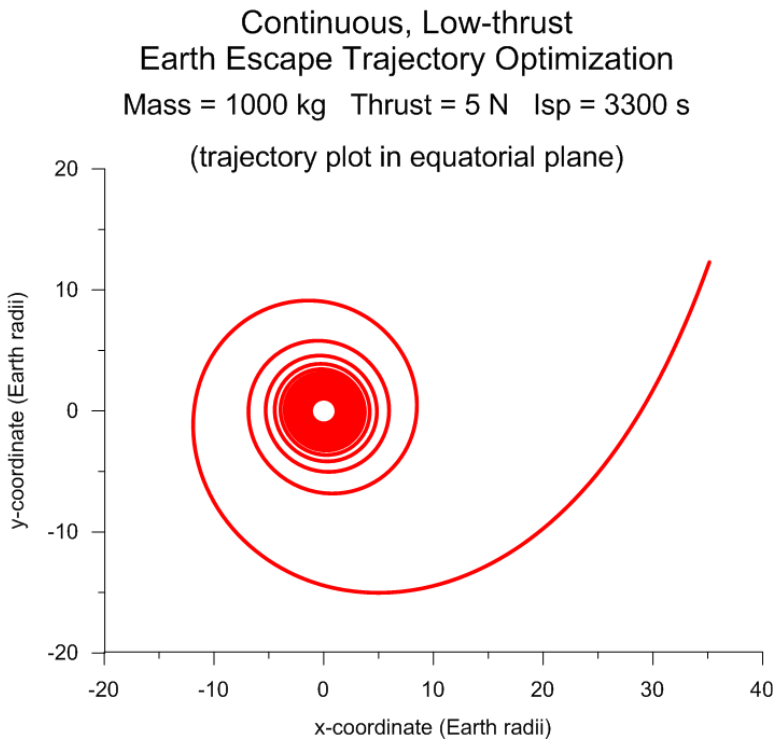


This next plot illustrates the behavior of the flight path angle and specific orbital energy during the propulsive maneuver. As expected the final orbital energy is zero.





This final plot is a graphics display of the transfer trajectory in the x-y (equatorial) plane. The measure of the axes is Earth radii (ER).



Verification of the optimal control solution

The optimal control solution determined by the *Sparse Optimization Suite* can be verified by numerically integrating the orbital equations of motion with the *SOS*-computed initial park orbit conditions and the optimal control solution. This is equivalent to solving an initial value problem (IVP) that uses the optimal unit thrust vector solution. This part of the `escape_sos` computer program uses a Runge-Kutta-Fehlberg 7(8) variable step size method to integrate the orbital equations of motion.

The following is a display of the final solution computed using this *explicit* numerical integration method.

```
========================================
verification of optimal control solution
========================================

------------------
end of finite burn
------------------

     sma (km)              eccentricity       inclination (deg)       argper (deg)
 -.327318028673D+13    0.100000004129D+01    0.999999562784D+01    0.310802143576D+03

    raan (deg)          true anomaly (deg)       arglat (deg)          period (min)
 0.346517444155D+03    0.823718554154D+02    0.331739989912D+02    0.000000000000D+00

     rx (km)                ry (km)              rz (km)              rmag (km)
 0.224185238198D+06    0.784688551678D+05    0.226712245563D+05    0.238600852220D+06

     vx (kps)              vy (kps)              vz (kps)              vmag (kps)
 0.663345695780D+00    0.167401640243D+01    0.314309877524D+00    0.182788104833D+01

final mass              818.781528480988      kilograms

propellant mass         181.218471519012      kilograms
```

```
thrust duration         325.810122682355      hours
                        13.5754217784315      days

delta-v                 6470.38206445709      meters/second

specific energy       1.844226993341636E-010 km**2/sec**2
```

Creating an initial guess

The `escape_sos` software implements a simple numerical method for estimating the total maneuver time. The program simply integrates the equations of motion until the specific orbital energy becomes positive. The software uses a tangential thrusting steering method to generate an initial guess for the optimal trajectory. For tangential thrusting, the unit thrust vector in the modified equinoctial frame at all times is simply $\mathbf{u}_T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$. Please note that this type of steering method creates a *coplanar* initial guess. It works best when the initial and final orbits are nearly coplanar.

The dynamic variables at each grid point of the initial guess are determined by setting the initial guess option `INIT(1) = 6` with `INIT(2) = 2` within the `odeinp` subroutine for this aerospace trajectory optimization problem. These program options create an initial guess from the numerical integration of the equations of motion coded in the `oderhs` subroutine. The `INIT(1) = 6` program option tells the *Sparse Optimization Suite* to construct an initial guess by solving an initial value problem (IVP) with a linear control approximation. The `INIT(2) = 2` program option tells the program to use the Dormand-Prince variable step size numerical method to solve the initial value problem.

Binary restart data files can also be used to initialize an `escape_sos` simulation. A typical scenario is

1) Create a binary restart file from a converged and optimized simulation
2) Modify the original input file with slightly different spacecraft characteristics, propulsive parameters, or perhaps final mission targets and/or constraints
3) Use the previously created binary restart file as the initial guess for the new simulation

This technique works well provided the two simulations are not dramatically different. Sometimes it may be necessary to make successive small changes in the mission definition and run multiples simulations to eventually reach the final desired solution.

Problem setup

This section provides additional details about the software implementation. It explains such things as point and path constraints, the performance index and the numerical technique used to create an initial guess for the software.

Initial orbit constraints

The lower and upper bounds for all modified equinoctial elements are set equal to the initial user-defined modified equinoctial orbital elements (subscript *i*) as follows:

$$p_L = p_U = p_i \qquad f_L = f_U = f_i \qquad g_L = g_U = g_i$$

$$h_L = h_U = h_i \qquad k_L = k_U = k_i \qquad L_L = L_U = L_i$$

The initial time is also constrained to a value of zero. In optimal control terminology, these equality constraints or boundary conditions are called *point functions*.

Performance index – minimize transfer time

The objective function or performance index *J* for this simulation is the total maneuver time. This is simply $J = t_f$.

Since this simulation involves a single continuous maneuver, this is equivalent to minimizing the required propellant mass. The value of the `maxmin` indicator tells the software whether the user is minimizing or maximizing the performance index.

Path constraint – unit thrust vector scalar magnitude

The scalar magnitude of the components of the unit thrust vector at any time during the simulation is constrained as follows

$$\left|\mathbf{u}_T\right| = \sqrt{u_{T_r}^2 + u_{T_t}^2 + u_{T_n}^2} = 1$$

This formulation avoids numerical problems that may occur when using thrust steering angles as control variables.

Point functions – final mission orbit constraints

The final, user-defined orbit inclination is constrained by enforcing

$$\sqrt{h^2 + k^2} = \tan\left(\frac{i}{2}\right)$$

where *i* is the mission orbit inclination.

The final, user-defined specific orbital energy C_3 is constrained with the following calculation

$$C_3 = v^2 - 2\frac{\mu}{r}$$

where *v* is the scalar velocity, *r* is the geocentric distance and μ is the Earth's gravitational constant, all evaluated at the final condition.

Bounds on the dynamic variables

The following lower and upper bounds are applied to the modified equinoctial orbital elements *during* the orbital transfer.

$$10\,p_f \le p \le 0.9\,p_i$$

$$-1 \le f \le +1 \qquad -1 \le g \le +1$$

$$-1 \le h \le +1 \qquad -1 \le k \le +1$$

The true longitude at any time is unbounded. The spacecraft mass at the initial time is fixed to the user-defined initial value.

Finally, the three components of the unit thrust vector are constrained as follows:

$$-1.1 \le u_r \le +1.1 \qquad -1.1 \le u_t \le +1.1 \qquad -1.1 \le u_n \le +1.1$$

Technical discussion

In this computer program, the orbital motion of the spacecraft is modeled in the modified equinoctial orbital elements coordinate system. Please consult *Appendix A – Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System* for a definition of these orbital elements along with the equations of motion in this system.

Propulsive Thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)} \hat{\mathbf{u}}_T$$

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_r} & u_{T_t} & u_{T_n} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system. *The components of this unit vector are the control variables.*

The propellant mass flow rate is determined from

$$\dot{m} = \frac{dm}{dt} = \frac{T}{g\, I_{sp}}$$

where g is the acceleration of gravity and I_{sp} is the specific impulse of the propulsive system. The product $g\, I_{sp}$ is also called the *exhaust velocity*. The spacecraft mass at any mission elapsed time t is given by $m(t) = m_{sc_i} - \dot{m}\, t$ where m_{sc_i} is the initial mass of the spacecraft and $\dot{m}$ is the propellant flow rate.

The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as follows

$$u_{T_r} = \sin\theta \qquad u_{T_t} = \cos\theta\cos\psi \qquad u_{T_n} = \cos\theta\sin\psi$$

Finally, the pitch and yaw angles can be determined from the components of the unit thrust vector according to

$$\theta = \sin^{-1}\left(u_{T_r}\right) \qquad \psi = \tan^{-1}\left(u_{T_n}, u_{T_t}\right)$$

Both steering angles are defined with respect to a local-vertical, local-horizontal (LVLH) system located at the spacecraft. The in-plane pitch angle is positive above the "local horizontal" and the out-of-plane yaw angle is positive in the direction of the angular momentum vector. The inverse tangent calculation in the second equation is a four-quadrant operation.

The `escape_sos` software provides the steering angles and the components of the unit thrust vector in both the inertial and modified equinoctial coordinate systems. The inertial-to/from-modified equinoctial coordinate transformations is described in Appendix A.

In the `escape_sos` computer program, the components of the inertial unit thrust vector are defined in terms of the right ascension α and the declination angle δ as follows

$$u_{T_{ECI_x}} = \cos\alpha\cos\delta \qquad u_{T_{ECI_y}} = \sin\alpha\cos\delta \qquad u_{T_{ECI_z}} = \sin\delta$$

Finally, the right ascension and declination angles can be determined from the components of the ECI unit thrust vector according to

$$\alpha = \tan^{-1}\left(u_{T_{ECI_y}}, u_{T_{ECI_x}}\right) \qquad \delta = \sin^{-1}\left(u_{T_{ECI_z}}\right)$$

where the calculation for right ascension is a four-quadrant inverse tangent operation.

Contents of the simulation summary and csv files

This section is a summary of the information contained in the simulation summary screen displays and the CSV data files produced by the `escape_sos` software.

The simulation summary screen display contains the following information:

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (min) = orbital period in minutes

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (km/sec) = x-component of the spacecraft's velocity vector in kilometers per second

vy (km/sec) = y-component of the spacecraft's velocity vector in kilometers per second

vz (km/sec) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

final mass = final spacecraft mass in kilograms

propellant mass = expended propellant mass in kilograms

thrust duration = maneuver duration in seconds

delta-v = scalar magnitude of the maneuver in meters/seconds

The delta-v is determined using a cubic spline integration of the thrust acceleration data at each collocation node.

The comma-separated-variable disk file is created by the `odeprt` Fortran subroutine and contains the following information.

time (hrs) = simulation time since ignition in hours

time (days) = simulation time since ignition in days

semimajor axis = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination = orbital inclination in degrees

argument of perigee = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

period (min) = orbital period in minutes

mass (kg) = spacecraft mass in kilograms

thracc = thrust acceleration in meters/second squared

yaw (deg) = thrust vector yaw angle in degrees

pitch (deg) = thrust vector pitch angle in degrees

rasc (deg) = thrust vector inertial right ascension angle in degrees

decl (deg) = thrust vector inertial declination angle in degrees

perigee altitude = perigee altitude in kilometers

apogee altitude = apogee altitude in kilometers

ut-radial = radial component of unit thrust vector

ut-tangential = tangential component of unit thrust vector

ut-normal = normal component of unit thrust vector

semi-parameter = orbital semiparameter in kilometers

f equinoctial element = modified equinoctial orbital element

g equinoctial element = modified equinoctial orbital element

h equinoctial element = modified equinoctial orbital element
k equinoctial element = modified equinoctial orbital element

true longitude = true longitude in degrees

rx (er) = x-component of the spacecraft's position vector in Earth radii

ry (er) = y-component of the spacecraft's position vector in Earth radii

rz (er) = z-component of the spacecraft's position vector in Earth radii

rmag (er) = spacecraft's geocentric distance in Earth radii

fpa (deg) = flight path angle in degrees

energy = specific orbital energy in (kilometers per second) squared

"Technique for Escape from Geosynchronous Transfer Orbit Using a Solar Sail", Victoria L. Coverstone and John E. Prussing, AIAA *Journal of Guidance, Control and Dynamics*, Vol. 26, No. 4, July-August 2003.

"Optimal Nonplanar Escape from Circular Orbits", T. N. Edelbaum, AIAA *Journal*, Vol. 9, No. 12.

"Optimal Low-Thrust Takeoff from an Orbit About an Oblate Planet", Robert A. Jacobson and William F. Powers, *Celestial Mechanics*, **15** (1977) 161-189.

Appendix A

Trajectory Modeling and Targeting in the Modified Equinoctial Orbital Elements System

The modified equinoctial orbital elements (MEE) are a set of orbital elements that are useful for trajectory analysis and optimization. They are valid for circular, elliptic, and hyperbolic orbits. This feature is important for trajectories that may transition to and from elliptic and hyperbolic orbits. These equations exhibit no singularity for zero eccentricity and orbital inclinations equal to 0 and 90 degrees. However, two components of the orbital element set are singular for an orbital inclination of 180^0.

The relationship between direct modified equinoctial and classical orbital elements is defined by the following definitions:

$$p = a\left(1-e^2\right) \qquad f = e\cos\left(\omega+\Omega\right) \qquad g = e\sin\left(\omega+\Omega\right)$$

$$h = \tan\left(i/2\right)\cos\Omega \qquad k = \tan\left(i/2\right)\sin\Omega \qquad L = \Omega+\omega+\theta$$

where

$p =$ semiparameter
$a =$ semimajor axis
$e =$ orbital eccentricity
$i =$ orbital inclination
$\omega =$ argument of periapsis
$\Omega =$ right ascension of the ascending node
$\theta =$ true anomaly
$L =$ true longitude

The relationship between classical and modified equinoctial orbital elements is summarized as follows:

semimajor axis

$$a = \frac{p}{1-f^2-g^2}$$

orbital eccentricity

$$e = \sqrt{f^2+g^2}$$

orbital inclination

$$i = 2\tan^{-1}\left(\sqrt{h^2+k^2}\right)$$

argument of periapsis

$$\omega = \tan^{-1}\left(g, f\right) - \tan^{-1}\left(k, h\right)$$

$$\sin\omega = \frac{g\,h - f\,k}{e\tan\left(i/2\right)} \qquad \cos\omega = \frac{f\,h + g\,k}{e\tan\left(i/2\right)}$$

right ascension of the ascending node (RAAN)

$$\Omega = \tan^{-1}(k,h)$$

$$\sin\Omega = \frac{k}{\tan(i/2)} \qquad \cos\Omega = \frac{h}{\tan(i/2)}$$

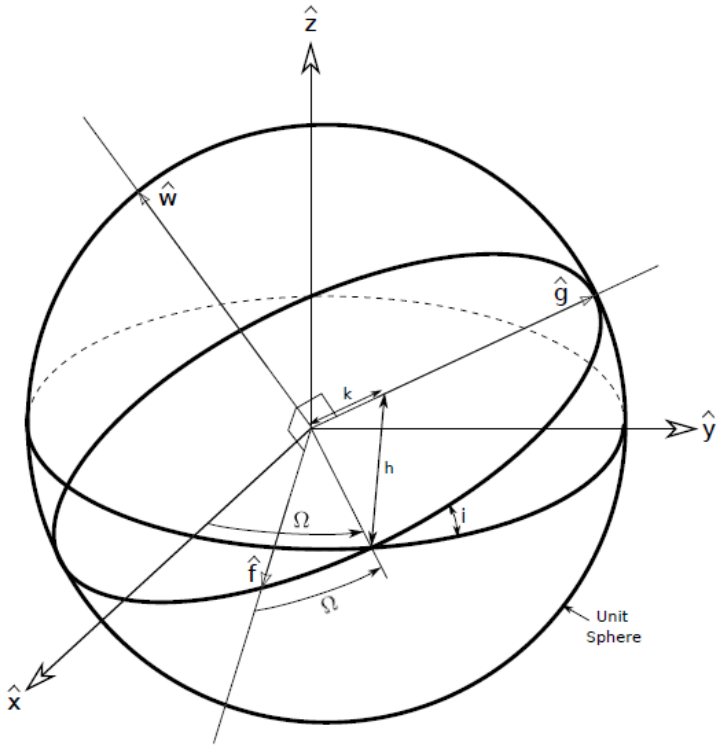
true anomaly

$$\theta = L - (\omega + \Omega) = L - \tan^{-1}(g, f)$$

$$\sin\theta = \frac{1}{e}(f\sin L - g\cos L) \qquad \cos\theta = \frac{1}{e}(f\cos L + g\sin L)$$

In these expressions, an inverse tangent expression of the form $\theta = \tan^{-1}(a,b)$ denotes a four quadrant evaluation where $a = \sin\theta$ and $b = \cos\theta$.

The following figure illustrates the orientation of the Earth-centered-inertial (ECI) and MEE coordinate systems on a unit sphere. The unit vectors in the ECI system are $\hat{x}, \hat{y}, \hat{z}$ and the unit vectors in the MEE system are $\hat{f}, \hat{g}, \hat{w}$.



$$\hat{\mathbf{f}} = \begin{Bmatrix} 1-k^2+h^2 \\ 2kh \\ 2k \end{Bmatrix} \qquad \hat{\mathbf{g}} = \begin{Bmatrix} 2kh \\ 1+k^2-h^2 \\ 2h \end{Bmatrix}$$

The mathematical relationships between an inertial state vector and the corresponding modified equinoctial elements are summarized as follows:

position vector

$$\mathbf{r} = \begin{bmatrix} \frac{r}{s^2}\left(\cos L + \alpha^2 \cos L + 2hk \sin L\right) \\ \frac{r}{s^2}\left(\sin L - \alpha^2 \sin L + 2hk \cos L\right) \\ \frac{2r}{s^2}\left(h \sin L - k \cos L\right) \end{bmatrix}$$

velocity vector

$$\mathbf{v} = \begin{bmatrix} -\frac{1}{s^2}\sqrt{\frac{\mu}{p}}\left(\sin L + \alpha^2 \sin L - 2hk \cos L + g - 2fhk + \alpha^2 g\right) \\ -\frac{1}{s^2}\sqrt{\frac{\mu}{p}}\left(-\cos L + \alpha^2 \cos L + 2hk \sin L - f + 2ghk + \alpha^2 f\right) \\ \frac{2}{s^2}\sqrt{\frac{\mu}{p}}\left(h \cos L + k \sin L + fh + gk\right) \end{bmatrix}$$

where

$$\alpha^2 = h^2 - k^2 \qquad s^2 = 1 + h^2 + k^2 \qquad r = \frac{p}{w} \qquad w = 1 + f \cos L + g \sin L$$

Equations of orbital motion

The system of nonlinear, first-order modified equinoctial equations of orbital motion with respect to time is given by the next six equations.

$$\dot{p} = \frac{dp}{dt} = \frac{2p}{w}\sqrt{\frac{p}{\mu}}\Delta_t$$

$$\dot{f} = \frac{df}{dt} = \sqrt{\frac{p}{\mu}}\left[\Delta_r \sin L + \left[\left(w+1\right)\cos L + f\right]\frac{\Delta_t}{w} - \left(h \sin L - k \cos L\right)\frac{g\Delta_n}{w}\right]$$

$$\dot{g} = \frac{dg}{dt} = \sqrt{\frac{p}{\mu}}\left[-\Delta_r \cos L + \left[\left(w+1\right)\sin L + g\right]\frac{\Delta_t}{w} + \left(h \sin L - k \cos L\right)\frac{f\Delta_n}{w}\right]$$

$$\dot{h} = \frac{dh}{dt} = \sqrt{\frac{p}{\mu}}\frac{s^2\Delta_n}{2w}\cos L \qquad \dot{k} = \frac{dk}{dt} = \sqrt{\frac{p}{\mu}}\frac{s^2\Delta_n}{2w}\sin L$$

$$\dot{L} = \frac{dL}{dt} = \sqrt{\mu p}\left(\frac{w}{p}\right)^2 + \frac{1}{w}\sqrt{\frac{p}{\mu}}\left(h \sin L - k \cos L\right)\Delta_n$$

where $\Delta_r, \Delta_t, \Delta_n$ are *non-two-body* perturbations in the radial, tangential and normal directions, respectively. For an Earth orbiting spacecraft, the radial direction is along the geocentric radius vector of the spacecraft measured positive in a direction away from the gravitational center, the tangential direction is perpendicular to this radius vector measured positive in the direction of orbital motion, and the normal direction is positive along the angular momentum vector of the spacecraft's orbit.

The equations of orbital motion can also be expressed in vector form as

$$\dot{\mathbf{y}} = \frac{d\mathbf{y}}{dt} = \mathbf{A}(\mathbf{y})\mathbf{P} + \mathbf{b}$$

where

$$\mathbf{A} = \begin{Bmatrix} 0 & \frac{2p}{w}\sqrt{\frac{p}{\mu}} & 0 \\ \sqrt{\frac{p}{\mu}}\sin L & \sqrt{\frac{p}{\mu}}\frac{1}{w}\left[(w+1)\cos L + f\right] & -\sqrt{\frac{p}{\mu}}\frac{g}{w}\left[h\sin L - k\cos L\right] \\ -\sqrt{\frac{p}{\mu}}\cos L & \sqrt{\frac{p}{\mu}}\frac{1}{w}\left[(w+1)\sin L + g\right] & \sqrt{\frac{p}{\mu}}\frac{f}{w}\left[h\sin L - k\cos L\right] \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{s^2\cos L}{2w} \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{s^2\sin L}{2w} \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{1}{w}\left[h\sin L - k\cos L\right] \end{Bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \sqrt{\mu p}\left(\frac{w}{p}\right)^2 \end{bmatrix}^T.$$

The <u>total</u> non-two-body acceleration vector is given by

$$\mathbf{P} = \Delta_r \hat{\mathbf{i}}_r + \Delta_t \hat{\mathbf{i}}_t + \Delta_n \hat{\mathbf{i}}_n$$

where $\hat{\mathbf{i}}_r, \hat{\mathbf{i}}_t$ and $\hat{\mathbf{i}}_n$ are unit vectors in the radial, tangential and normal directions. These unit vectors can be computed from the inertial position vector **r** and velocity vector **v** according to

$$\hat{\mathbf{i}}_r = \frac{\mathbf{r}}{|\mathbf{r}|} \qquad \hat{\mathbf{i}}_n = \frac{\mathbf{r}\times\mathbf{v}}{|\mathbf{r}\times\mathbf{v}|} \qquad \hat{\mathbf{i}}_t = \hat{\mathbf{i}}_n \times \hat{\mathbf{i}}_r = \frac{(\mathbf{r}\times\mathbf{v})\times\mathbf{r}}{|\mathbf{r}\times\mathbf{v}||\mathbf{r}|}$$

For *unperturbed* two-body motion, $\mathbf{P} = 0$ and the first five equations of motion are simply $\dot{p} = \dot{f} = \dot{g} = \dot{h} = \dot{k} = 0$. Therefore, for two-body motion these modified equinoctial orbital elements are constant. The true longitude is often called the *fast variable* of this orbital element set.

The dynamical equations of motion with true longitude L as the independent variable can be determined from the following two vector expressions:

$$\dot{\mathbf{z}} = \frac{d\mathbf{z}}{dL}\frac{dL}{dt} = \mathbf{f}(\mathbf{z})$$

$$\frac{d\mathbf{z}}{dL} = \mathbf{f}(\mathbf{z})\left[\frac{dL}{dt}\right]^{-1} = \frac{1}{\dot{L}}\mathbf{f}(\mathbf{z})$$

where $\mathbf{z}$ is the vector of modified equinoctial orbital elements.

Propulsive thrust

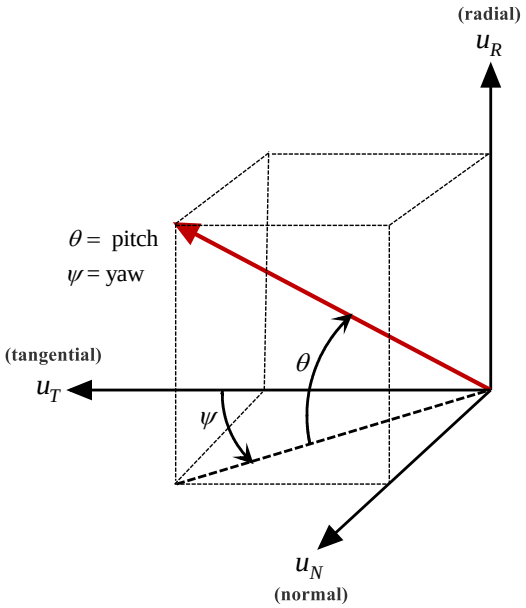
The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \eta\frac{T}{m}\hat{\mathbf{u}}$$

where T is the thrust magnitude, m is the spacecraft mass, $\hat{\mathbf{u}} = \begin{bmatrix} u_r & u_t & u_n \end{bmatrix}$ is the unit pointing thrust vector expressed in the spacecraft-centered radial-tangential-normal coordinate system, and η is the throttle setting. The components of the unit thrust vector can also be defined in terms of the in-plane pitch angle θ and the out-of-plane yaw angle ψ as follows:

$$u_r = \sin\theta \qquad u_t = \cos\theta\cos\psi \qquad u_n = \cos\theta\sin\psi$$

The pitch angle is positive above the "local horizontal" and the yaw angle is positive in the direction of the angular momentum vector.



The relationship between a unit thrust vector in the ECI coordinate system $\hat{\mathbf{u}}_{T_{ECI}}$ and the corresponding unit thrust vector in the modified equinoctial system $\hat{\mathbf{u}}_{T_{MEE}}$ is given by

$$\hat{\mathbf{u}}_{T_{ECI}} = \begin{bmatrix} \hat{\mathbf{i}}_r & \hat{\mathbf{i}}_t & \hat{\mathbf{i}}_n \end{bmatrix}\hat{\mathbf{u}}_{T_{MEE}}$$

This relationship can also be expressed as

$$\hat{\mathbf{u}}_{T_{ECI}} = [\mathbf{Q}]\hat{\mathbf{u}}_{T_{MEE}} = \begin{bmatrix} \hat{\mathbf{r}}_x & \left(\hat{\mathbf{h}}\times\hat{\mathbf{r}}\right)_x & \hat{\mathbf{h}}_x \\ \hat{\mathbf{r}}_y & \left(\hat{\mathbf{h}}\times\hat{\mathbf{r}}\right)_y & \hat{\mathbf{h}}_y \\ \hat{\mathbf{r}}_z & \left(\hat{\mathbf{h}}\times\hat{\mathbf{r}}\right)_z & \hat{\mathbf{h}}_z \end{bmatrix} \hat{\mathbf{u}}_{T_{MEE}}$$

Finally, the transformation of the unit thrust vector in the ECI system to the modified equinoctial coordinate system is given by $\hat{\mathbf{u}}_{T_{MEE}} = [\mathbf{Q}]^T \hat{\mathbf{u}}_{T_{ECI}}$.

For the case of *tangential steering*

$$\hat{\mathbf{u}}_{T_{ECI}} = \left[\left(\hat{\mathbf{h}}\times\hat{\mathbf{r}}\right)_x \quad \left(\hat{\mathbf{h}}\times\hat{\mathbf{r}}\right)_y \quad \left(\hat{\mathbf{h}}\times\hat{\mathbf{r}}\right)_z\right]^T.$$

Non-spherical Earth gravity

The non-spherical gravitational acceleration vector can be expressed as

$$\mathbf{g} = g_N \hat{\mathbf{i}}_N - g_r \hat{\mathbf{i}}_r$$

where

$$\hat{\mathbf{i}}_N = \frac{\hat{\mathbf{e}}_N - \left(\hat{\mathbf{e}}_N^T \hat{\mathbf{i}}_r\right)\hat{\mathbf{i}}_r}{\left\|\hat{\mathbf{e}}_N - \left(\hat{\mathbf{e}}_N^T \hat{\mathbf{i}}_r\right)\hat{\mathbf{i}}_r\right\|}$$

and

$$\hat{\mathbf{e}}_N = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

In these equations the north direction component is indicated by subscript *N* and the radial direction component is subscript *r*.

The contributions due to a *zonal* gravity model of order *n* are as follows:

$$g_N = -\frac{\mu\cos\phi}{r^2}\sum_{k=2}^{n}\left(\frac{R_e}{r}\right)^k P_k' J_k$$

$$g_r = -\frac{\mu}{r^2}\sum_{k=2}^{n}(k+1)\left(\frac{R_e}{r}\right)^k P_k J_k$$

where

$\mu =$ gravitational constant
$r =$ geocentric distance of the spacecraft
$R_e =$ equatorial radius of the Earth
$\phi =$ geocentric latitude
$J_k =$ zonal gravity coefficient
$P_k =$ kth order Legendre polynomial

For a *zonal only* Earth gravity model, the east component is identically zero.

Finally, the zonal gravity perturbation contribution is $\mathbf{a}_g = \mathbf{Q}^T\mathbf{g}$ where $\mathbf{Q} = \left[\hat{\mathbf{i}}_r \quad \hat{\mathbf{i}}_t \quad \hat{\mathbf{i}}_n\right]$.

For J_2 effects only, the three components are as follows:

$$\Delta_{J_{2_r}} = -\frac{3\mu J_2 R_e^2}{2r^4}\left[1 - \frac{12\left(h\sin L - k\cos L\right)^2}{\left(1+h^2+k^2\right)^2}\right]$$

$$\Delta_{J_{2_t}} = -\frac{12\mu J_2 R_e^2}{r^4}\left[\frac{\left(h\sin L - k\cos L\right)\left(h\cos L + k\sin L\right)}{\left(1+h^2+k^2\right)^2}\right]$$

$$\Delta_{J_{2_n}} = -\frac{6\mu J_2 R_e^2}{r^4}\left[\frac{\left(1-h^2-k^2\right)\left(h\sin L - k\cos L\right)}{\left(1+h^2+k^2\right)^2}\right]$$

Planetary perturbations

The general vector equation for *point-mass* perturbations such as the Moon or planets is given by

$$\ddot{\mathbf{r}} = -\sum_{j=1}^{n}\mu_j\left[\frac{\mathbf{d}_j}{d_j^3} + \frac{\mathbf{s}_j}{s_j^3}\right]$$

In this equation, $\mathbf{s}_j$ is the vector from the primary body to the secondary body *j*, μ_j is the gravitational constant of the secondary body and $\mathbf{d}_j = \mathbf{r} - \mathbf{s}_j$, where **r** is the position vector of the spacecraft relative to the primary body.

To avoid numerical problems, use is made of Professor Richard Battin's $F(q)$ function given by

$$F(q_k) = q_k\left[\frac{3 + 3q_k + q_k^2}{1 + \left(\sqrt{1+q_k}\right)^3}\right]$$

where

$$q_k = \frac{\mathbf{r}^T\left(\mathbf{r} - 2\mathbf{s}_k\right)}{\mathbf{s}_k^T\mathbf{s}_k}$$

The acceleration due to other planets can now be expressed as

$$\ddot{\mathbf{r}} = -\sum_{k=1}^{n}\frac{\mu_k}{d_k^3}\left[\mathbf{r} + F(q_k)\mathbf{s}_k\right]$$

Finally, the perturbation due to secondary bodies in the modified equinoctial coordinate system is given by $\mathbf{a} = \left[\mathbf{Q}\right]^T\mathbf{t}$ where $\mathbf{Q} = \left[\hat{\mathbf{i}}_r \quad \hat{\mathbf{i}}_t \quad \hat{\mathbf{i}}_n\right]$.

Targeting with modified equinoctial elements

This section describes techniques for "targeting" with modified equinoctial orbital elements by enforcing equality and inequality mission constraints during the trajectory optimization. Constraint formulations that enforce both the sine and cosine of a desired orbital element should be used whenever possible. This approach involves a combination of equality and inequality constraints and ensures that the "targeted" orbital element is in the correct quadrant.

To illustrate this technique, here are several examples for different values of argument of perigee and the corresponding mission constraints.

$$0^\circ < \omega < 90^\circ \rightarrow \begin{cases} \sin\omega > 0 \rightarrow gh - fk > 0 \\ f\,h + g\,k = e\tan(i/2)\cos\omega \end{cases}$$

$$\omega = 270^\circ \rightarrow \begin{cases} \sin\omega \le 0 \rightarrow gh - fk \le 0 \\ \cos\omega = 0 \rightarrow f\,h + g\,k = 0 \end{cases}$$

$$\omega = 178^\circ \rightarrow \begin{cases} g\,h - f\,k = e\tan(i/2)\sin\omega \\ \cos\omega \le 0 \rightarrow f\,h + g\,k \le 0 \end{cases}$$

The following is a *sign* table of the sine and cosine for each quadrant.

quadrant	sine	cosine
1	+	+
2	+	–
3	–	–
4	–	+

orbital eccentricity constraint

$$e = \sqrt{f^2 + g^2}$$

For a <u>circular</u> orbit, $f = g = 0$.

<u>orbital inclination constraint</u>

$$\tan\left(\frac{i}{2}\right) = \sqrt{h^2 + k^2}$$

For an <u>equatorial</u> orbit, $h = k = 0$.

<u>argument of perigee constraints</u>

$$g\,h - f\,k = e\sin\omega\tan(i/2) \rightarrow \sin\omega = \frac{g\,h - f\,k}{e\tan(i/2)}$$

$$f\,h + g\,k = e\cos\omega\tan(i/2) \rightarrow \cos\omega = \frac{f\,h + g\,k}{e\tan(i/2)}$$

right ascension of the ascending node constraints

$$k = \tan(i/2)\sin\Omega \rightarrow \sin\Omega = \frac{k}{\tan(i/2)}$$

$$h = \tan(i/2)\cos\Omega \rightarrow \cos\Omega = \frac{h}{\tan(i/2)}$$

true anomaly constraints

$$\theta = L - (\omega + \Omega) = L - \tan^{-1}(g, f)$$

In general,

$$\sin\theta = \frac{1}{e}(f\sin L - g\cos L) \qquad \cos\theta = \frac{1}{e}(f\cos L + g\sin L)$$

For a circular orbit,

$$\sin\theta = \sin L\cos\Omega - \cos L\sin\Omega \qquad \cos\theta = \cos L\cos\Omega + \sin L\sin\Omega$$

For a circular, equatorial orbit,

$$\theta = L \qquad \sin\theta = \sin L \qquad \cos\theta = \cos L$$

Targeting example

For a user-defined semimajor axis, orbital eccentricity and inclination, the set of modified equinoctial equality constraints are as follows:

$$p = \tilde{p} \qquad \sqrt{f^2 + g^2} = \tilde{e} \qquad \sqrt{h^2 + k^2} = \tan(\tilde{i}/2)$$

where the tilde indicates the value of the user-defined classical orbital element.

"On the Equinoctial Orbital Elements", R. A. Brouke and P. J. Cefola, *Celestial Mechanics*, Vol. 5, pp. 303-310, 1972.

"A Set of Modified Equinoctial Orbital Elements", M. J. H. Walker, B. Ireland and J. Owens, *Celestial Mechanics*, Vol. 36, pp. 409-419, 1985.

"Equinoctial Orbit Elements: Application to Optimal Transfer Problems", Jean A. Kechichian, AIAA 90-2976, AIAA/AAS Astrodynamics Conference, Portland, OR, 20-22 August 1990.

Appendix B

Trajectory Modeling in the Flight Path System

This appendix summarizes the equations of motion in the flight path coordinate system. It also provides the relationship between flight path coordinates and inertial position and velocity vectors. Flight path coordinates are typically used to model *endospheric* flight and inertial coordinates are used to model *exospheric* flight mechanics.

The first-order flight path equations of motion with respect to time and relative to a rotating spherical Earth are as follows:

geocentric radius

$$\dot{r} = \frac{dr}{dt} = V \sin\gamma$$

geographic longitude

$$\dot{\lambda} = \frac{d\lambda}{dt} = V \frac{\cos\gamma \sin\psi}{r \cos\delta}$$

geocentric declination

$$\dot{\delta} = \frac{d\delta}{dt} = V \frac{\cos\gamma \cos\psi}{r}$$

speed

$$\dot{V} = \frac{dV}{dt} = \frac{\left(T\cos\alpha - D\right)}{m} - g_r \sin\gamma + g_\phi \cos\gamma \cos\psi$$
$$+ \omega_e^2 r \cos\delta \left(\sin\gamma \cos\delta - \sin\delta \cos\gamma \cos\psi\right)$$

flight path angle

$$\dot{\gamma} = \frac{d\gamma}{dt} = \frac{V}{r}\cos\gamma + \left(\frac{T\sin\alpha + L}{mV}\right)\cos\beta - \frac{g_r \cos\gamma}{V} - \frac{g_\phi \sin\gamma \cos\psi}{V}$$
$$+ 2\omega_e \sin\psi \cos\delta + \omega_e^2 \frac{r}{V}\cos\delta \left(\cos\psi \sin\gamma \sin\delta + \cos\gamma \cos\delta\right)$$

flight azimuth

$$\dot{\psi} = \frac{d\psi}{dt} = \frac{V}{r}\tan\delta \sin\psi \cos\gamma + \left(\frac{T\sin\alpha + L}{mV\cos\gamma}\right)\sin\beta - \frac{\sin\psi}{V\cos\gamma} g_\phi$$
$$+ 2\omega_e \left(\sin\delta - \cos\psi \cos\delta \tan\gamma\right) + \frac{r}{V\cos\gamma}\omega_e^2 \sin\psi \cos\delta \sin\delta$$

where

$r =$ geocentric radius
$V =$ speed
$\gamma =$ flight path angle
$\delta =$ geocentric declination
$\lambda =$ longitude $(+ \text{ east})$
$\psi =$ flight azimuth $(+ \text{ clockwise from north})$
$\beta =$ bank angle $(+ \text{ for a right turn})$
$\alpha =$ angle-of-attack
$r_e =$ Earth equatorial radius
$\omega_e =$ Earth inertial rotation rate
$g_r =$ radial component of gravity
$g_\phi =$ latitudinal component of gravity
$L =$ aerodynamic lift force
$D =$ aerodynamic drag force
$T =$ propulsive thrust
$m =$ spacecraft mass
$C_L =$ lift coefficient (non-dimensional)
$C_D =$ drag coefficient (non-dimensional)
$S =$ aerodynamic reference area
$\rho =$ atmospheric density $= f(h)$
$h =$ altitude

Earth gravity

The components of the Earth's gravity vector in the flight path coordinate system can be determined from the gradient of the potential function according to

$$\mathbf{F}_G = \nabla U = \begin{Bmatrix} \frac{1}{r}\frac{\partial U}{\partial \phi} \\ 0 \\ -\frac{\partial U}{\partial r} \end{Bmatrix} = \begin{Bmatrix} g_\phi \\ 0 \\ g_r \end{Bmatrix}$$

where

$$U = \frac{\mu}{r}\left[1 - \sum_{j=1}^{\infty}\left(\frac{r_e}{r}\right)^j J_j P_{j0}(\sin\phi)\right]$$

$$\frac{\partial U}{\partial r} = \frac{\mu}{r^2}\left[-1 + \sum_{l=2}^{\infty}(l+1)\left(\frac{r_e}{r}\right)^l J_l P_{l0}(\sin\phi)\right] \qquad \frac{1}{r}\frac{\partial U}{\partial \phi} = -\frac{\mu}{r^2}\sum_{l=2}^{\infty}\left(\frac{r_e}{r}\right)^l J_l \frac{\partial P_{l0}}{\partial \phi}$$

$$P_{j0}\left(\sin\phi\right)=\sum_{t=0}^{k}T_{jt}\sin^{j-2t}\phi \qquad T_{jt}=\left(-1\right)^{t}\left(2j-2t\right)!/\,2lt!\left(j-t\right)!\left(j-2t\right)!$$

For a *zonal-only* gravity model of order four, the Legendre functions and their partial derivatives are given by

$$P_{20}=\frac{1}{2}\left(3\sin^{2}\phi-1\right) \qquad P_{30}=\frac{1}{2}\left(5\sin^{3}\phi-3\sin\phi\right) \qquad P_{40}=\frac{1}{8}\left(35\sin^{4}\phi-30\sin^{2}\phi+3\right)$$

$$\frac{\partial P_{20}}{\partial\phi}=3\sin\phi\cos\phi \qquad \frac{\partial P_{30}}{\partial\phi}=\frac{3}{2}\left(5\sin^{2}\phi-1\right)\cos\phi \qquad \frac{\partial P_{40}}{\partial\phi}=\frac{5}{2}\left(7\sin^{2}\phi-3\right)\sin\phi\cos\phi$$

Converting an ECI state vector to flight path coordinates

The transformation of an ECI position vector $\mathbf{r}_{eci}$ to an ECF position vector $\mathbf{r}_{ecf}$ is given by the following vector-matrix operation

$$\mathbf{r}_{ECF}=\left[\mathbf{T}\right]\mathbf{r}_{ECI}$$

where the elements of the transformation matrix $\left[\mathbf{T}\right]$ are given by

$$\left[\mathbf{T}\right]=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$

and θ is the Greenwich apparent sidereal time at the time of interest. Greenwich apparent sidereal time is given by the following expression

$$\theta=\theta_{g0}+\omega_{e}t$$

where θ_{g0} is the Greenwich apparent sidereal time at 0 hours UTC, ω_{e} is the inertial rotation rate of the Earth, and *t* is the elapsed time since 0 hours UTC.

Finally, the flight path coordinates are determined from the following set of equations

$$r=\sqrt{r_{ECF_x}^{2}+r_{ECF_y}^{2}+r_{ECF_z}^{2}} \qquad v=\sqrt{v_{ECF_x}^{2}+v_{ECF_y}^{2}+v_{ECF_z}^{2}}$$

$$\lambda=\tan^{-1}\left(r_{ECF_y},r_{ECF_x}\right) \qquad \delta=\sin^{-1}\left(\frac{r_{ECF_z}}{\left|\mathbf{r}_{ECF}\right|}\right)$$

$$\gamma=\sin^{-1}\left(-\frac{v_{r_z}}{\left|\mathbf{v}_{r}\right|}\right) \qquad \psi=\tan^{-1}\left[v_{r_y},v_{r_x}\right]$$

where

$$\mathbf{v}_{r}=\begin{bmatrix}-\sin\delta\cos\lambda & -\sin\delta\sin\lambda & \cos\delta\\ -\sin\lambda & \cos\lambda & 0\\ -\cos\delta\cos\lambda & -\cos\delta\sin\lambda & -\sin\delta\end{bmatrix}\mathbf{v}_{ECF}$$

Converting flight path coordinates to an ECI state vector

This coordinate conversion is often necessary to determine the orbital inclination and other trajectory characteristics at entrance or exit from a celestial body atmosphere.

The Earth-centered-fixed (ECF) position and velocity vectors of the vehicle can be determined from the flight path coordinates with the following set of equations:

$$r_{ECF_x} = r\cos\delta\cos\lambda \qquad r_{ECF_y} = r\cos\delta\sin\lambda \qquad r_{ECF_z} = r\sin\delta$$

$$v_{ECF_x} = v\left[\cos\lambda\left(-\cos\psi\sin\beta\sin\delta + \cos\beta\cos\delta\right) - \sin\psi\sin\beta\sin\lambda\right]$$

$$v_{ECF_y} = v\left[\sin\lambda\left(-\cos\psi\sin\beta\sin\delta + \cos\beta\cos\delta\right) + \sin\psi\sin\beta\cos\lambda\right]$$

$$v_{ECF_z} = v\left(\cos\psi\cos\delta\sin\beta + \cos\beta\cos\delta\right)$$

where $\beta = 90^\circ - \gamma$.

The transformation from ECF to ECI coordinates involves the transpose of the ECI-to-ECF transformation matrices described above as follows

$$\mathbf{r}_{ECI} = \left[\mathbf{T}\right]^T \mathbf{r}_{ECF}$$

$$\mathbf{v}_{ECI} = \left[\mathbf{T}\right]^T \dot{\mathbf{r}}_{ECF} + \left[\dot{\mathbf{T}}\right]^T \mathbf{r}_{ECF} = \left[\mathbf{T}\right]^T \mathbf{v}_{ECF} + \left[\dot{\mathbf{T}}\right]^T \mathbf{r}_{ECF}$$

The ECF velocity vector is determined by differentiating this expression

$$\mathbf{v}_{ECF} = \left[\mathbf{T}\right]\dot{\mathbf{r}}_{ECI} + \left[\dot{\mathbf{T}}\right]\mathbf{r}_{ECI} = \left[\mathbf{T}\right]\mathbf{v}_{ECI} + \left[\dot{\mathbf{T}}\right]\mathbf{r}_{ECI}$$

The elements of the $\left[\dot{\mathbf{T}}\right]$ matrix are as follows

$$\left[\dot{\mathbf{T}}\right] = \begin{bmatrix} -\omega_e\sin\theta & \omega_e\cos\theta & 0 \\ -\omega_e\cos\theta & -\omega_e\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Vehicle aerodynamics

The next section is a summary of useful equations related to general aerodynamic characteristics of aerospace vehicles.

general form of a drag polar

$$C_D = C_{D_0} + k\left|C_L\right|^n$$

lift-to-drag ratio

$$E = \frac{L}{D} = \frac{C_L}{C_D} = \frac{C_L}{C_{D_0} + kC_L^n}$$

maximum lift-to-drag ratio (value of E at which $dE/dC_L = 0$)

$$E^* = \frac{\left(C_{D_0} + kC_L^n\right) - C_L\left(nkC_L^{n-1}\right)}{\left(C_{D_0} + kC_L^n\right)^2}$$

lift coefficient at maximum lift-to-drag ratio

$$C_L^* = \sqrt[n]{\frac{C_{D_0}}{k\left(n-1\right)}}$$

drag coefficient at maximum lift-to-drag ratio

$$C_D^* = \frac{nC_{D_0}}{\left(n-1\right)}$$

In general,

$$E^* = \frac{C_L^*}{C_D^*} = \frac{\sqrt[n]{\left(n-1\right)^{n-1}}}{n\sqrt[n]{kC_{D_0}^{n-1}}}$$

For a *parabolic* drag polar $\left(n = 2\right)$,

$$C_D = C_{D_0} + kC_L^2$$

where

$C_D =$ drag coefficient

$C_{D_0} =$ drag coefficient at 0° angle-of-attack

$C_L =$ lift coefficient

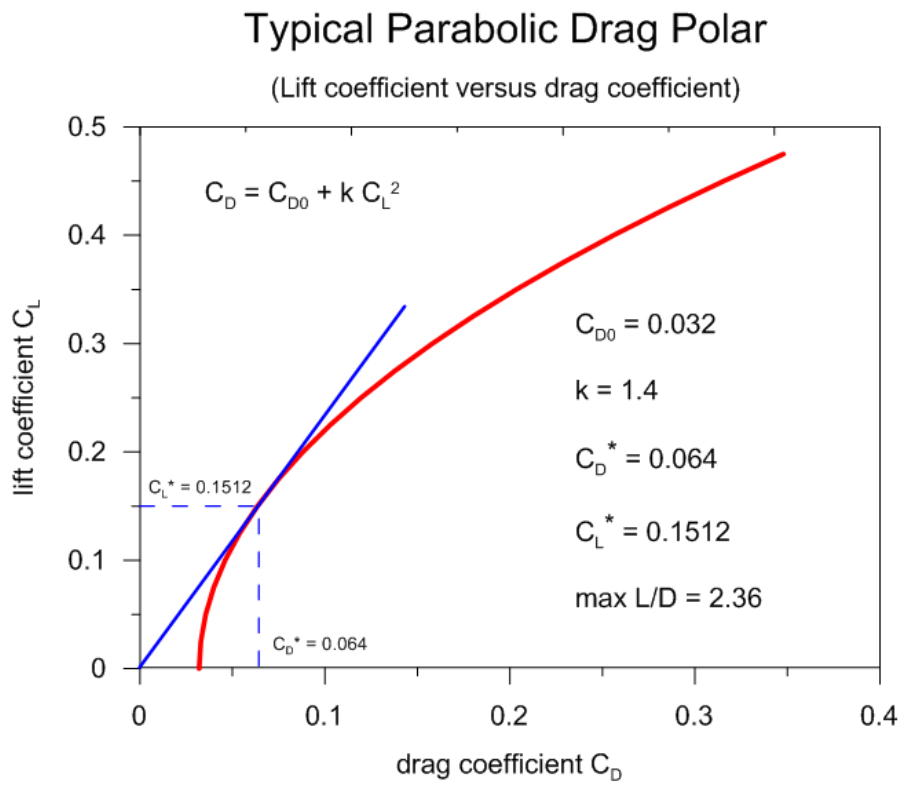
$k =$ constant

$C_D^* = 2C_{D_0} =$ drag coefficient at maximum L/D

$C_L^* = \sqrt{C_{D_0} / k} =$ lift coefficient at maximum L/D

$$E^* = \left(\frac{C_L}{C_D}\right)_{\max} = \frac{1}{2\sqrt{kC_{D_0}}} = \text{maximum L/D}$$

The following is a graphic display of a typical parabolic drag polar (shown by the red curve). The blue line is tangent to the drag polar and visually helps locate the lift and drag coefficients at maximum lift over drag (max L/D).



Heat rate and heat load

The heat rate experienced by an aerospace vehicle can be computed using Chapman's stagnation point heat rate equation according to

$$\dot{q} = \frac{dq}{dt} = \frac{17{,}600}{\sqrt{R_N}}\left(\frac{V}{V_0}\right)^{3.15}\sqrt{\frac{\rho}{\rho_0}} \qquad \left(\frac{\text{BTU}}{\text{ft}^2 - \text{sec}}\right)$$

where

R_N = nose radius

V = relative velocity at the spacecraft location

V_0 = "local circular velocity" at the Earth's surface $= \sqrt{\mu / r_e}$

ρ = atmospheric density at the spacecraft location

ρ_0 = atmospheric density at the Earth's surface

μ = gravitational constant of the Earth

r_e = radius of the Earth

The accumulated heat load at any mission elapsed time is determined from the integral of heat rate according to

$$q_L = \int_{t_1}^{t_2} \dot{q}(t)\,dt$$

The lift L and drag D forces are given by

$$L = \frac{1}{2}\rho V^2 C_L A \qquad D = \frac{1}{2}\rho V^2 C_D A$$

where ρ is the atmospheric density and A is the aerodynamic reference area. In these expressions, $\frac{1}{2}\rho V^2$ is the dynamic pressure.

Optimal Trajectories in Atmospheric Flight, N. X. Vinh, Elsevier, 1981.

Hypersonic and Planetary Flight Mechanics, N. X. Vinh, R. D. Culp, and A. Busemann, University of Michigan Press, 1980.

Appendix C

Cartesian Equations of Motion

The vector system of second-order, nonlinear differential equations of spacecraft orbital motion are given by

$$\mathbf{a}(\mathbf{r},\mathbf{v},t)=\ddot{\mathbf{r}}(\mathbf{r},\dot{\mathbf{r}},t)=\mathbf{a}_g(\mathbf{r},t)+\mathbf{a}_m(\mathbf{r},t)+\mathbf{a}_s(\mathbf{r},t)+\mathbf{a}_d(\mathbf{r},\mathbf{v},t)+\mathbf{a}_{srp}(\mathbf{r},t)+\mathbf{a}_T(\mathbf{r},t)$$

where

t = dynamical time
$\mathbf{r}$ = inertial position vector of the spacecraft
$\mathbf{v}$ = inertial velocity vector of the spacecraft
$\mathbf{a}_g$ = acceleration due to the Earth's gravity
$\mathbf{a}_m$ = acceleration due to the Moon
$\mathbf{a}_s$ = acceleration due to the Sun
$\mathbf{a}_d$ = acceleration due to aerodynamic drag
$\mathbf{a}_{srp}$ = acceleration due to solar radiation pressure
$\mathbf{a}_T$ = acceleration due to propulsive thrust

Acceleration due to non-spherical Earth gravity

A spherical harmonic representation of the Earth's geopotential function is given by

$$\Phi(r,\phi,\lambda)=\frac{\mu}{r}+\frac{\mu}{r}\sum_{n=1}^{\infty}C_n^0\left(\frac{R}{r}\right)^n P_n^0(u)+\frac{\mu}{r}\sum_{n=1}^{\infty}\sum_{m=1}^{n}\left(\frac{R}{r}\right)^n P_n^m(u)\left[S_n^m\sin m\lambda+C_n^m\cos m\lambda\right]$$

where ϕ is the geocentric latitude of the spacecraft, λ is the geocentric east longitude of the spacecraft and $r=|\mathbf{r}|=\sqrt{x^2+y^2+z^2}$ is the geocentric distance of the spacecraft. In this expression the *S's* and *C's* are *unnormalized* harmonic coefficients of the geopotential, and the *P's* are associated Legendre polynomials of degree *n* and order *m* with argument $u = \sin\phi$.

The software calculates the spacecraft's acceleration due to the Earth's gravity field with a vector equation derived from the gradient of the potential function expressed as $\mathbf{a}_g(\mathbf{r},t)=\nabla\Phi(\mathbf{r},t)$.

This acceleration vector is a combination of pure two-body or *point mass* gravity acceleration and the gravitational acceleration due to higher order non-spherical terms in the Earth's geopotential. In terms of the Earth's geopotential Φ, the inertial rectangular cartesian components of the spacecraft's acceleration vector are as follows

$$\ddot{x}=\left(\frac{1}{r}\frac{\partial\Phi}{\partial r}-\frac{z}{r^2\sqrt{x^2+y^2}}\frac{\partial\Phi}{\partial\phi}\right)x-\left(\frac{1}{x^2+y^2}\frac{\partial\Phi}{\partial\lambda}\right)y$$

$$\ddot{y} = \left(\frac{1}{r} \frac{\partial \Phi}{\partial r} - \frac{z}{r^2 \sqrt{x^2 + y^2}} \frac{\partial \Phi}{\partial \phi} \right) y + \left(\frac{1}{x^2 + y^2} \frac{\partial \Phi}{\partial \lambda} \right) x$$

$$\ddot{z} = \left(\frac{1}{r} \frac{\partial \Phi}{\partial r} \right) z + \left(\frac{\sqrt{x^2 + y^2}}{r^2} \frac{\partial \Phi}{\partial \phi} \right)$$

The three partial derivatives of the geopotential with respect to r, ϕ, λ are given by

$$\frac{\partial \Phi}{\partial r} = -\frac{1}{r} \left(\frac{\mu}{r} \right) \sum_{n=2}^{N} \left(\frac{R}{r} \right)^n (n+1) \sum_{m=0}^{n} \left(C_n^m \cos m\lambda + S_n^m \sin m\lambda \right) P_n^m (\sin \phi)$$

$$\frac{\partial \Phi}{\partial \phi} = \left(\frac{\mu}{r} \right) \sum_{n=2}^{N} \left(\frac{R}{r} \right)^n \sum_{m=0}^{n} \left(C_n^m \cos m\lambda + S_n^m \sin m\lambda \right) \left[P_n^{m+1} (\sin \phi) - m \tan \phi \, P_n^m (\sin \phi) \right]$$

$$\frac{\partial \Phi}{\partial \lambda} = \left(\frac{\mu}{r} \right) \sum_{n=2}^{N} \left(\frac{R}{r} \right)^n \sum_{m=0}^{n} m \left(S_n^m \cos m\lambda - C_n^m \sin m\lambda \right) P_n^m (\sin \phi)$$

where

$R =$ radius of the Earth

$r =$ geocentric distance of the spacecraft

$S_n^m, C_n^m =$ harmonic coefficients

$\phi =$ geocentric latitude of the spacecraft $= \sin^{-1}(z/r)$

$\lambda =$ longitude of the spacecraft $= \alpha - \alpha_g$

$\alpha =$ right ascension of the spacecraft $= \tan^{-1}(y/x)$

$\alpha_g =$ right ascension of Greenwich

Right ascension is measure positive east of the vernal equinox, longitude is measured positive east of Greenwich, and latitude is positive above the Earth's equator and negative below.

For $m = 0$, the coefficients are called *zonal* terms, when $m = n$ the coefficients are *sectorial* terms, and for $n > m \neq 0$ the coefficients are called *tesseral* terms.

The Legendre polynomials with argument $\sin \phi$ are computed using recursion relationships given by

$$P_n^0 (\sin \phi) = \frac{1}{n} \left[(2n-1) \sin \phi \, P_{n-1}^0 (\sin \phi) - (n-1) P_{n-2}^0 (\sin \phi) \right]$$

$$P_n^n (\sin \phi) = (2n-1) \cos \phi \, P_{n-1}^{n-1} (\sin \phi), \qquad m \neq 0, \; m < n$$

$$P_n^m (\sin \phi) = P_{n-2}^m (\sin \phi) + (2n-1) \cos \phi \, P_{n-1}^{m-1} (\sin \phi), \qquad m \neq 0, \; m = n$$

where the first few associated Legendre functions are given by

$$P_0^0 (\sin \phi) = 1, \quad P_1^0 (\sin \phi) = \sin \phi, \quad P_1^1 (\sin \phi) = \cos \phi$$

and $P_i^j = 0$ for $j > i$.

The trigonometric arguments are determined from expansions given by

$$\sin m\lambda = 2\cos\lambda \sin(m-1)\lambda - \sin(m-2)\lambda$$

$$\cos m\lambda = 2\cos\lambda \cos(m-1)\lambda - \cos(m-2)\lambda$$

$$m\tan\phi = (m-1)\tan\phi + \tan\phi$$

Acceleration due to the point-mass gravity of the sun and moon

The acceleration contribution of the moon represented by a *point mass* is given by

$$\mathbf{a}_m(\mathbf{r},t) = -\mu_m\left(\frac{\mathbf{r}_{m-b}}{|\mathbf{r}_{m-b}|^3} + \frac{\mathbf{r}_{e-m}}{|\mathbf{r}_{e-m}|^3}\right)$$

where

$\mu_m =$ gravitational constant of the moon

$\mathbf{r}_{m-b} =$ position vector from the moon to the satellite

$\mathbf{r}_{e-m} =$ position vector from the Earth to the moon

The acceleration contribution of the sun represented by a *point mass* is given by

$$\mathbf{a}_s(\mathbf{r},t) = -\mu_s\left(\frac{\mathbf{r}_{s-b}}{|\mathbf{r}_{s-b}|^3} + \frac{\mathbf{r}_{e-s}}{|\mathbf{r}_{e-s}|^3}\right)$$

where

$\mu_s =$ gravitational constant of the sun

$\mathbf{r}_{s-b} =$ position vector from the sun to the spacecraft

$\mathbf{r}_{e-s} =$ position vector from the Earth to the sun

To avoid numerical problems, use is made of Professor Richard Battin's $f(q)$ function given by

$$f(q_k) = q_k\left[\frac{3+3q_k+q_k^2}{1+\left(\sqrt{1+q_k}\right)^3}\right]$$

where

$$q_k = \frac{\mathbf{r}^T(\mathbf{r}-2\mathbf{s}_k)}{\mathbf{s}_k^T\mathbf{s}_k}$$

The point-mass acceleration due to *n* gravitational bodies can now be expressed as

$$\ddot{\mathbf{r}} = -\sum_{k=1}^{n}\frac{\mu_k}{d_k^3}\left[\mathbf{r} + f(q_k)\mathbf{s}_k\right]$$

In these equations, $\mathbf{s}_k$ is the vector from the primary body to the secondary body, μ_k is the gravitational constant of the secondary body and $\mathbf{d}_k = \mathbf{r} - \mathbf{s}_k$, where $\mathbf{r}$ is the position vector of the spacecraft relative

to the primary body. The derivation of the $f(q)$ functions is described in Section 8.4 of "An *Introduction to the Mathematics and Methods of Astrodynamics*, Revised Edition", by Richard H. Battin, AIAA Education Series, 1999.

Acceleration due to atmospheric drag

The acceleration experienced by a spacecraft due to atmospheric drag is computed using the following vector expression

$$\mathbf{a}_d(\mathbf{r},\mathbf{v},t) = -\frac{1}{2}\rho(\mathbf{r},t)|\mathbf{v}_r|\mathbf{v}_r\frac{C_d A}{m}$$

where

$\mathbf{v}_r =$ satellite velocity vector relative to the atmosphere
$\rho =$ atmospheric density
$C_d =$ drag coefficient of the spacecraft
$A =$ reference area of the spacecraft
$m =$ mass of the spacecraft

The aerodynamic drag algorithm assumes that the atmosphere rotates at the same angular speed as the Earth. With this reasonable assumption the spacecraft's relative velocity vector is given by $\mathbf{v}_r = \mathbf{v} - \mathbf{w} \times \mathbf{r}$ where $\mathbf{w}$ is the inertial rotation vector of the Earth. The angular velocity vector of the Earth is $\mathbf{w} = \omega_e \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ where $\omega_e = 7.292115486\text{E-}5$ radians per second is the scalar inertial rotation rate.

The cross-product expansion of the previous equation gives the three components of the relative velocity vector as follows

$$\mathbf{v}_r = \begin{bmatrix} v_x + \omega_e r_y \\ v_y - \omega_e r_x \\ v_z \end{bmatrix}$$

Acceleration due to solar radiation pressure

We can define a *solar radiation constant* for any spacecraft as a function of its size, mass and surface reflective properties according to the equation

$$C_{srp} = \gamma P_s a^2 \frac{A}{m}$$

where

$\gamma =$ reflectivity constant
$P_s =$ solar radiation constant
$a =$ astronomical unit
$A =$ surface area normal to the incident radiation
$m =$ mass of the spacecraft

The reflectivity constant is a dimensionless number between 0 and 2. For a perfectly absorbent body $\gamma = 1$, for a perfectly reflective body $\gamma = 2$, and for a translucent body $\gamma < 1$. For example, the reflectivity constant for an aluminum surface is approximately 1.96.

The value of the solar radiation pressure on a perfectly absorbing spacecraft surface at a distance of one Astronomical Unit from the Sun is

$$P_s = \frac{1358}{c} \frac{watts}{m^2}$$

where *c* is the speed of light in meters/second.

The acceleration vector of the spacecraft due to solar radiation pressure is given by:

$$\mathbf{a}_{srp} = C_{srp} \frac{\mathbf{r}_{sat-to-Sun}}{r^3_{sat-to-Sun}}$$

where

$\mathbf{r}_{sat-to-Sun} = \mathbf{r}_{sat} - \mathbf{r}_{Earth-to-Sun}$

$\mathbf{r}_{sat}$ = geocentric, inertial position vector of the spacecraft

$\mathbf{r}_{Earth-to-Sun}$ = geocentric, inertial position vector of the sun

During the numerical integration process, the software must determine if the spacecraft is in Earth shadow or sunlight. Obviously, there can be no solar radiation perturbation during Earth eclipse of the spacecraft orbit. The software makes use of a *shadow parameter* to determine eclipse conditions. This parameter is defined by the following expression

$$\varphi = -\frac{\left|\mathbf{r}_{sc} \times \mathbf{r}_{es}\right|}{\left|\mathbf{r}_{es}\right|} sign(\mathbf{r}_{sc} \bullet \mathbf{r}_{es})$$

where $\mathbf{r}_{sc}$ is the geocentric, inertial position vector of the spacecraft and $\mathbf{r}_{es}$ is the geocentric, inertial position vector of the Sun relative to the spacecraft.

The *critical* values of the shadow parameter for the penumbra (subscript *p*) and umbra part (subscript *u*) of the shadow are given by the following formulas

$$\varphi_p = \left|\mathbf{r}_{sc}\right| \sin\psi_p \qquad \varphi_u = \left|\mathbf{r}_{sc}\right| \sin\psi_u$$

The penumbra and umbra shadow angles are found from

$$\psi_p = \eta + \theta_p \qquad \psi_u = \eta - \theta_u$$

These are the angles between the geocentric anti-Sun vector and the geocentric vector to a spacecraft at the time of shadow entrance or exit.

If we represent the shadow as a cylinder, the shadow angle is given by $\eta = \sin^{-1}\left(r_e / r_{sc}\right)$.

The corresponding penumbra and umbra *cone* angles are as follows

$$\theta_p = \sin^{-1}\left(\frac{r_s + r_e}{r_{es}}\right) \qquad \theta_u = \sin^{-1}\left(\frac{r_s - r_e}{r_{es}}\right)$$

where

$r_e =$ radius of the Earth

$r_s =$ radius of the Sun

$r_{es} =$ distance from the Earth to the Sun

If the condition $\varphi_u < \varphi \le \varphi_p$ is true, the spacecraft is in the penumbra part of the Earth's shadow, and if the inequality $0 \le \varphi \le \varphi_u$ is true, the spacecraft is in the umbra part of the shadow. If the absolute value of the shadow parameter is larger than the penumbra value, the spacecraft is in full sunlight.

The shadow calculations used in the *Computer Methods for Aerospace Trajectory Optimization* software also assume that the Earth's atmosphere increases the radius of the Earth by two percent.

Acceleration due to propulsive thrust

The acceleration due to propulsive thrust can be expressed as

$$\mathbf{a}_T = \frac{T}{m(t)}\hat{\mathbf{u}}_T$$

where T is the thrust magnitude, m is the spacecraft mass and $\hat{\mathbf{u}}_T = \begin{bmatrix} u_{T_x} & u_{T_y} & u_{T_z} \end{bmatrix}^T$ is the unit pointing thrust vector expressed in the spacecraft-centered cartesian coordinate system.

An Introduction to the Mathematics and Methods of Astrodynamics, Richard H. Battin, AIAA Education Series, 1987.

Spacecraft Mission Design, Charles D. Brown, AIAA Education Series, 1992.

Analytical Mechanics of Space Systems, Hanspeter Schaub and John L. Junkins, AIAA Education Series, 2003.

Modern Astrodynamics, Victor R. Bond and Mark C. Allman, Princeton University Press, 1996.

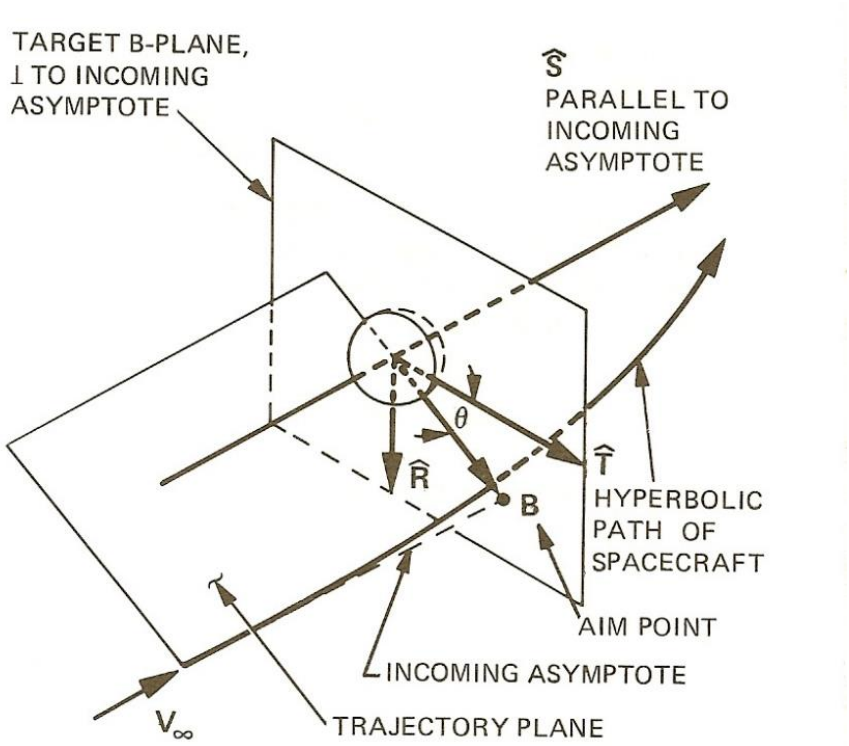
Orbital Mechanics, Vladimir A. Chobotov, AIAA Education Series, 2002.

Hypersonic and Planetary Entry Flight Mechanics, Vinh, Busemann and Culp, The University of Michigan Press, 1980.

Appendix D

B-Plane Geometry, Coordinates and Targeting

The derivation of B-plane coordinates is described in the classic JPL reports, "A Method of Describing Miss Distances for Lunar and Interplanetary Trajectories" and "Some Orbital Elements Useful in Space Trajectory Calculations", both by William Kizner. The following diagram illustrates the geometry of the B-plane coordinate system.



The arrival asymptote unit vector $\hat{\mathbf{S}}$ is given by

$$\hat{\mathbf{S}} = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

where δ_∞ and α_∞ are the declination and right ascension of the asymptote of the incoming hyperbola at the arrival planet.

B-plane calculations

This section describes the conversion of inertial coordinates of an arrival or departure hyperbola to fundamental B-plane coordinates and vectors.

angular momentum vector

$$\mathbf{h} = \mathbf{r} \times \mathbf{v}$$

radius rate

$$\dot{r} = \mathbf{r} \bullet \mathbf{v} / |\mathbf{r}|$$

semi-parameter

$$p = \frac{h^2}{\mu}$$

semimajor axis

$$a = \frac{r}{\left(2 - \frac{r v^2}{\mu}\right)}$$

orbital eccentricity

$$e = \sqrt{1 - p/a}$$

true anomaly

$$\cos\nu = \frac{p - r}{e\,r} \qquad \sin\nu = \frac{\dot{r}\,h}{e\mu}$$

B-plane magnitude

$$B = r_p\sqrt{1 + \frac{2\mu}{r_p V_\infty^2}} = \frac{\mu}{V_\infty^2}\sqrt{\left(1 + V_\infty^2\frac{r_p}{\mu}\right)^2 - 1}$$

fundamental vectors

$$\hat{\mathbf{z}} = \frac{r\,\mathbf{v} - \dot{r}\,\mathbf{r}}{h}$$

$$\hat{\mathbf{p}} = \cos\theta\,\hat{\mathbf{r}} - \sin\theta\,\hat{\mathbf{z}} \qquad \hat{\mathbf{q}} = \sin\theta\,\hat{\mathbf{r}} + \cos\theta\,\hat{\mathbf{z}}$$

S vector

$$\mathbf{S} = -\frac{a}{\sqrt{a^2 + b^2}}\hat{\mathbf{p}} + \frac{b}{\sqrt{a^2 + b^2}}\hat{\mathbf{q}} \qquad \text{where } b = \sqrt{p|a|}$$

B vector

$$\mathbf{B} = \frac{b^2}{\sqrt{a^2 + b^2}}\hat{\mathbf{p}} + \frac{ab}{\sqrt{a^2 + b^2}}\hat{\mathbf{q}}$$

T vector

$$\mathbf{T} = \frac{\left(S_y^2, -S_x^2, 0\right)^T}{\sqrt{S_x^2 + S_y^2}}$$

R vector

$$\mathbf{R} = \mathbf{S} \times \mathbf{T} = \left(-S_z T_y,\ S_z T_x,\ S_x T_y - S_y T_x\right)^T$$

where **r** and **v** are the spacecraft inertial position and velocity vectors and μ is the planet's gravitational constant.

Techniques for B-Plane targeting

This section describes several techniques for using B-plane coordinates to *target* to specific planetary or moon encounter conditions. These targets are usually formulated as equality mission constraints which are enforced while solving the trajectory optimization problem.

user-defined B-plane coordinates

For this targeting option, the two nonlinear equality constraints enforced by the nonlinear programming (NLP) algorithm are

$$\left(\mathbf{B}\bullet\mathbf{T}\right)_p - \left(\mathbf{B}\bullet\mathbf{T}\right)_u = 0 \qquad \left(\mathbf{B}\bullet\mathbf{R}\right)_p - \left(\mathbf{B}\bullet\mathbf{R}\right)_u = 0$$

where the *p* subscript refers to coordinates predicted by the software and the *u* subscript denotes coordinates provided by the user. The *predicted* B-plane coordinates are based on the planet-centered flight conditions at closest approach.

targeting to user-defined entry interface (EI) conditions with user-defined inclination

For this targeting option, the following equations can be used to determine the required B-plane components based on the user-defined EI targets. These targets are the inertial flight path angle and altitude relative to a spherical planet model.

$$\mathbf{B}\bullet\mathbf{T} = b_t \cos\theta \qquad \mathbf{B}\bullet\mathbf{R} = b_t \sin\theta$$

where

$$b_t = \cos\gamma_{ei}\sqrt{\frac{2\mu r_{ei}}{v_\infty^2} + r_{ei}^2} \quad \text{and} \quad \cos\theta = \frac{\cos i}{\cos\delta_\infty}$$

Note that this targeting option could be modified to use a user-defined B-plane angle instead of orbital inclination at the entry interface.

In these equations, γ_{ei} is the user-defined flight path angle at the entry interface, r_{ei} is the arrival planet-centered radius at the entry interface (sum of planet equatorial radius plus user-defined EI altitude) and *i* is the user-defined orbital inclination.

Also, these dot product mission constraints can be expressed in terms of the *x*, *y* and *z* components of the spacecraft's inertial position and velocity vectors according to

$$\mathbf{B}\bullet\mathbf{T} = \frac{r_x v_y - r_y v_x}{\sqrt{v_x^2 + v_y^2}} \qquad \mathbf{B}\bullet\mathbf{R} = \frac{\left(r_x v_x + r_y v_y\right)v_z - \left(v_x^2 + v_y^2\right)r_z}{\sqrt{v_x^2 + v_y^2}\sqrt{v_x^2 + v_y^2 + v_z^2}}$$

targeting to a planet-centered grazing flyby with user-defined B-plane angle

The general expression for the periapsis radius of an encounter hyperbola at the arrival planet is given by

$$\tilde{r}_p = \frac{1}{\tilde{v}_\infty^2}\left(\sqrt{1 + \tilde{b}_\infty^2 \tilde{v}_\infty^4} - 1\right)$$

where the *normalized* quantities are

$\tilde{r}_p =$ normalized periapsis radius $= r_p / r_m$

$\tilde{b}_\infty =$ normalized b-plane magnitude $= b_\infty / r_m$

$\tilde{v}_\infty =$ normalized v-infinity speed $= v_\infty / v_{lc}$

$v_{lc} =$ local circular speed at Mars $= \sqrt{\mu_m / r_m}$

$r_m =$ radius of Mars

$\mu_m =$ gravitational constant of Mars

For a grazing flyby, $\tilde{r}_p = 1$ and the normalized B-plane distance or offset is equal to

$$\tilde{b}_\infty = \sqrt{1 + \frac{2}{\tilde{v}_\infty^2}}$$

Therefore, the required B-plane equality constraints are computed from

$$\mathbf{B} \bullet \mathbf{T} = b_\infty \cos\theta \qquad \mathbf{B} \bullet \mathbf{R} = b_\infty \sin\theta$$

where θ is the user-defined B-plane angle of the arrival trajectory. Please note that the B-plane angle is measured positive clockwise from the **T** axis of the B-plane coordinate system. The two equality constraints for this option are simply the difference between the predicted and required $\mathbf{B} \bullet \mathbf{T}$ and $\mathbf{B} \bullet \mathbf{R}$ components.

targeting to a planet-centered node/apse alignment trajectory

This targeting option determines when a spacecraft is simultaneously at a nodal crossing and periapsis of the arrival planet encounter hyperbola. The B-plane angle required for node-apse alignment is given by

$$\theta = \sin^{-1}\left(\frac{\sin\phi_h \hat{S}_z}{\cos\phi_h \sqrt{\hat{S}_x^2 + \hat{S}_y^2}} \right)$$

where $\phi_h = \sin^{-1}(1/e)$. This relationship is developed in the thesis of R. H. See.

Also noted in the thesis of R. H. See, the minimum and maximum values of orbital inclination can be determined from the *x* and *y* components of the unit $\hat{\mathbf{S}}$ vector according to

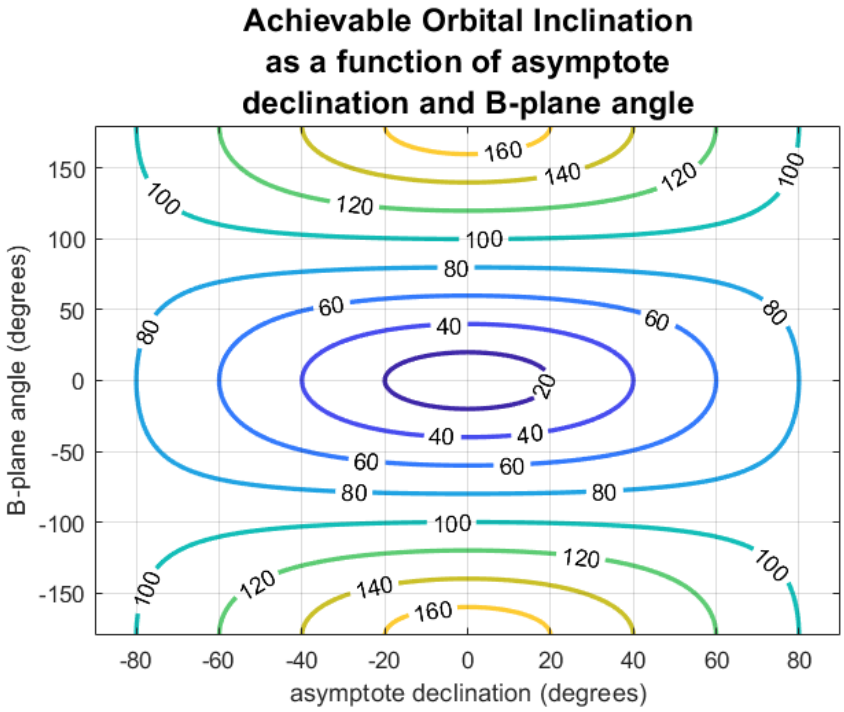
$$i_{\min} = \cos^{-1}\left(\sqrt{\hat{S}_x^2 + \hat{S}_y^2}\right) \qquad i_{\max} = \cos^{-1}\left(-\sqrt{\hat{S}_x^2 + \hat{S}_y^2}\right)$$

For an arrival hyperbola that lies in the equatorial plane, the following condition must be true

$$i_{\min} = 0 \rightarrow \sqrt{\hat{S}_x^2 + \hat{S}_y^2} = 1$$

Furthermore, if the *z* component of the incoming v-infinity vector is positive, $\mathbf{V}_{\infty_z} > 0$, an equatorial areocentric orbit is not possible.

The relationship between orbital inclination *i*, B-plane angle θ and asymptote declination δ of an incoming or outgoing hyperbola is given by $\cos i = \cos\theta \cos\delta$. The following is a contour plot illustrating the *achievable* inclination as a function of B-plane angle and declination.



Spacecraft Mission Design, Charles D. Brown, AIAA Education Series, 1992.

"Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers", John T. Betts, *The Journal of the Astronautical Sciences*, Vol. 41, No. 3, July-September 1993, pp. 349-371.

"A Complete and Fast Survey of the Orbital Insertion Design Space For Planetary Moon Missions", Robert Harding See, MS Thesis, University of Texas, August 2017.

"Trajectory correction maneuver design using an improved B-plane targeting method", D. Cho, Y. Chung and H. Bang, *Acta Astronautica*, 72 (2012) pp. 47-61.

Appendix E

Impulsive De-orbit from Earth Orbit

This appendix describes two analytic algorithms that can be used to compute the propulsive impulse required to de-orbit from circular and elliptical orbits. This impulsive answer can be used as an initial guess for finite-burn trajectory simulations.

Initial circular orbit

The scalar magnitude of the single impulsive maneuver required to de-orbit a spacecraft from an initial circular orbit can be determined from the following expression

$$\Delta V = V_{c_e}\sqrt{\frac{1}{\tilde{r}}}\left\{1-\sqrt{\frac{2\left(\tilde{r}-1\right)}{\left(\frac{\tilde{r}}{\cos\gamma_e}\right)^2-1}}\right\} = V_{c_i}\left\{1-\sqrt{\frac{2\left(\tilde{r}-1\right)}{\left(\frac{\tilde{r}}{\cos\gamma_e}\right)^2-1}}\right\}$$

where

$$\tilde{r} = \frac{h_i + r_{eq}}{h_e + r_{eq}} = \frac{r_i}{r_e} = \text{radius ratio}$$

$$V_{c_e} = \sqrt{\frac{\mu}{\left(h_e + r_{eq}\right)}} = \sqrt{\frac{\mu}{r_e}} = \text{ local circular velocity at entry interface}$$

$$V_{c_i} = \sqrt{\frac{\mu}{\left(h_i + r_{eq}\right)}} = \sqrt{\frac{\mu}{r_i}} = \text{ local circular velocity of initial circular orbit}$$

$\gamma_e =$ flight path angle at entry interface

$h_i =$ altitude of initial circular orbit

$h_e =$ altitude at entry interface

$r_i =$ radius of initial circular orbit

$r_e =$ radius at entry interface

$r_{eq} =$ Earth equatorial radius

$\mu =$ Earth gravitational constant

This algorithm is described in the technical article, “Deboost from Circular Orbits”, A. H. Milstead, *The Journal of the Astronautical Sciences*, Vol. XIII, No. 4, pp. 170-171, Jul-Aug., 1966. Additional information can be found in Chapter 5 of *Hypersonic and Planetary Entry Flight Mechanics* by Vinh, Busemann and Culp, The University of Michigan Press.

The true anomaly on the de-orbit trajectory at the entry interface θ_e can be determined from the following two equations

$$\sin\theta_e = \frac{\dot{r}}{e_d}\sqrt{\frac{a_d\left(1-e_d^2\right)}{\mu}} \qquad \cos\theta_e = \frac{a_d\left(1-e_d^2\right)}{e_d r_e} - \frac{1}{e_d}$$

and the following four-quadrant inverse tangent operation

$$\theta_e = \tan^{-1}\left(\sin\theta_e, \cos\theta_e\right)$$

where

$e_d =$ eccentricity of the de-orbit trajectory

$a_d =$ semimajor axis of the de-orbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu\left[2a_d r_e - r_e^2 - a_d^2\left(1-e_d^2\right)\right]}{a_d r_e^2}}$$

The elapsed time-of-flight between perigee of the de-orbit trajectory and the entry true anomaly θ_e is given by

$$t\left(\theta_e\right) = \frac{\tau}{2\pi}\left[2\tan^{-1}\left\{\sqrt{\frac{1-e_d}{1+e_d}}\tan\frac{\theta_e}{2}\right\} - \frac{e_d\sqrt{1-e_d^2}\sin\theta_e}{1+e_d\cos\theta_e}\right]$$

In this equation τ is the Keplerian orbital period of the de-orbit trajectory and is equal to $2\pi\sqrt{a_d^3/\mu}$.

Therefore, the flight time between the de-orbit impulse and entry interface is given by

$$\Delta t = t\left(\theta_e\right) - t\left(180^\circ\right) = t\left(\theta_e\right) - \frac{\tau}{2}$$

Finally, the orbital speed at the entry interface V_e can be determined from $V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$.

Initial elliptical orbit

The magnitude of the impulsive delta-v for de-orbit from an initial elliptical orbit is given by

$$\Delta V = \sqrt{\frac{\mu}{r_e}}\left(\sqrt{\frac{2\tilde{r}_p}{\tilde{r}_a\left(\tilde{r}_a+\tilde{r}_p\right)}} - \sqrt{\frac{2\left(\tilde{r}_a-1\right)}{\tilde{r}_a\left(\tilde{r}_a^2-\cos^2\gamma_e\right)}}\cos\gamma_e\right)$$

where

$r_e =$ geocentric radius at the entry altitude

$\tilde{r}_a = r_a / r_e$

$\tilde{r}_p = r_p / r_e$

$\gamma_e =$ flight path angle at entry

$r_a =$ apogee radius of the initial elliptical orbit

$r_p =$ perigee radius of the initial elliptical orbit

$\mu =$ planet gravitational constant

The true anomaly at entry can be determined from the following series of equations.

$$\sin\theta_e = \frac{\dot{r}}{e_d}\sqrt{\frac{a_d\left(1-e_d^2\right)}{\mu}} \qquad \cos\theta_e = \frac{a_d\left(1-e_d^2\right)}{e_d r_e} - \frac{1}{e_d} \qquad \theta_e = \tan^{-1}\left(\sin\theta_e, \cos\theta_e\right)$$

where

$e_d =$ eccentricity of the de-orbit trajectory

$a_d =$ semimajor axis of the de-orbit trajectory

$$\dot{r} = -\sqrt{\frac{\mu\left[2a_d r_e - r_e^2 - a_d^2\left(1-e_d^2\right)\right]}{a_d r_e^2}}$$

The time-of-flight between perigee and the entry interface true anomaly θ_e is given by

$$t\left(\theta_e\right) = \frac{\tau}{2\pi}\left[2\tan^{-1}\left\{\sqrt{\frac{1-e_d}{1+e_d}}\tan\frac{\theta_e}{2}\right\} - \frac{e_d\sqrt{1-e_d^2}\sin\theta_e}{1+e_d\cos\theta_e}\right]$$

In this equation, τ is the Keplerian or unperturbed orbital period of the de-orbit trajectory.

Therefore, the flight time between the de-orbit impulse time and the entry interface is given by

$$\Delta t = t\left(\theta_e\right) - t\left(180^\circ\right) = t\left(\theta_e\right) - \frac{\tau}{2}$$

Finally, the speed at the entry interface V_e can be determined from $V_e = \sqrt{\frac{2\mu}{r_e} - \frac{\mu}{a_d}}$.

The following is a typical numerical example of de-orbit from an initial circular orbit.

```
**************************************
single impulse deorbit from Earth orbits
**************************************

time and conditions prior to deorbit maneuver
---------------------------------------------
```

```
calendar date      18-Mar-2010

UTC time           12:30:45.875

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +6.87814000000000e+03  +0.00000000000000e+00  +2.85000000000000e+01  +1.00000000000000e+02

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +2.20000000000000e+02  +1.90000000000000e+02  +2.90000000000000e+02  +9.46163624134673e+01

        rx (km)                ry (km)               rz (km)               rmag (km)
 -5.45318321844679e+03  +2.83906883966968e+03  -3.08403806222734e+03  +6.87814000000000e+03

        vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 -4.00911535387506e+00  -6.35100857948859e+00  +1.24236145363939e+00  +7.61260651018449e+00
```

deorbit delta-v vector and magnitude

```
x-component of delta-v         80.301516  meters/second
y-component of delta-v        117.688815  meters/second
z-component of delta-v        -21.001409  meters/second
total delta-v                 144.014061  meters/second

deorbit delta-v pointing angles
-------------------------------

pitch angle                    -2.256618  degrees

yaw angle                    -179.973071  degrees

time and conditions after deorbit maneuver
------------------------------------------

calendar date      18-Mar-2010

UTC time           12:30:45.875

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +6.62986196765162e+03  +3.74561317348360e-02  +2.84998225494152e+01  +1.08881122818562e+02

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +2.20001021787282e+02  +1.81117979216534e+02  +2.89999102035096e+02  +8.95398701301721e+01

        rx (km)                ry (km)               rz (km)               rmag (km)
 -5.45318321844679e+03  +2.83906883966968e+03  -3.08403806222734e+03  +6.87814000000000e+03

        vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 -3.92881383778096e+00  -6.23331976439964e+00  +1.22136004444855e+00  +7.46870630131950e+00
```

time and conditions at entry interface

```
calendar date      18-Mar-2010

UTC time           13:00:49.638

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +6.63194303419306e+03  +3.87915541331080e-02  +2.85109989908368e+01  +1.07810489094078e+02


       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +2.19909866252810e+02  +2.99617139359573e+02  +4.74276284536510e+01  +8.95820323314157e+01

        rx (km)                ry (km)               rz (km)               rmag (km)
 -5.45318321844679e+03  +2.83906883966968e+03  -3.08403806222734e+03  +6.87814000000000e+03

        vx (kps)               vy (kps)              vz (kps)              vmag (kps)
 +7.50958358577267e+00  +3.73745768234497e-01  +2.46143688225353e+00  +7.91152343460458e+00
```

relative flight path coordinates at entry interface

```
east longitude             252.44489639  degrees
```

```
geocentric declination       20.58001385  degrees

flight path angle            -2.00000000  degrees

relative azimuth             68.65207490  degrees

position magnitude         6497.40258326  kilometers

velocity magnitude            7.49698673  kilometers/second

geodetic coordinates at entry interface
---------------------------------------

geodetic latitude            20.70458617  degrees

geodetic altitude           121.92000000  kilometers


flight time from maneuver to EI     30.06271094  minutes
```


"Nearly Circular Transfer Trajectories for Descending Satellites", George M. Low, NASA Technical Report R-3, 1959.

"Optimum Deboost Altitude for Specified Atmospheric Entry Angle", Jerome M. Baker, Bruce E. Baxter, and Paul D. Arthur, *AIAA Journal,* Vol. 1, No. 7, July 1963.

"Deboost from Circular Orbits", A. H. Milstead, *The Journal of the Astronautical Sciences,* Vol. XIII, No. 4, pp. 170-171, July-August, 1966.

"Geometric Theory of Optimum Disorbit Problems", A. Busemann and N. X. Vinh, NASA CR-750, April 1967.

"Autonomous Optimal Deorbit Targeting", Donald J. Jezewski, AAS 91-136, AAS/AIAA Spaceflight Mechanics Meeting, February 11-13, 1991.

"Algorithm for Prompt Prediction of the Trajectory of the Motion of a Descending Reentry Vehicle Gliding in the Rotating Atmosphere", V. V. Betanov, D. V. Doronin, and S. E. Zakharov, *Cosmic Research,* Vol. 37, No. 4, 1999, pp. 404-408.

"Analysis of the Accuracy of Ballistic Descent from a Circular Circumterrestrial Orbit", Yu. G. Sikharulidze and A. N. Korchagin, *Cosmic Research,* Vol. 40, No. 1, 2002, pp.75-87.

Appendix F

Computing an Asteroid or Comet Ephemeris

To model ballistic interplanetary missions involving asteroids and comets, the classical orbital elements of these types of small celestial bodies relative to the mean ecliptic and equinox of J2000 coordinate system must be provided by the user. These elements can be obtained from the JPL Near Earth Object (NEO) website (http://neo.jpl.nasa.gov).

These orbital elements consist of the following items

- calendar date of perihelion passage
- perihelion distance (astronomical units)
- orbital eccentricity (non-dimensional)
- orbital inclination (degrees)
- argument of perihelion (degrees)
- longitude of ascending node (degrees)

Here is a typical data display screen from the JPL NEO website.

Orbital Elements at Epoch 2459000.5 (2020-May-31.0) TDB
Reference: JPL 46 (heliocentric ecliptic J2000)

Element	Value	Uncertainty (1-sigma)	Units
e	.07755702331559071	4.807e-12	
a	2.767656854438644	1.0356e-11	au
q	2.553005627249392	1.947e-11	au
i	10.58862149177901	4.6132e-09	deg
node	80.28698636629558	6.1769e-08	deg
peri	73.73167094328808	6.6184e-08	deg
M	162.6862516879934	7.3559e-09	deg
t_0	2458240.497283818471 (2018-May-01.99728382)	3.6867e-08	TDB
period	1681.770739607881 4.60	9.4391e-09 2.584e-11	d yr
n	.2140600924499002	1.2014e-12	deg/d
Q	2.982308081627896	1.1159e-11	au

The software determines the mean anomaly of the asteroid or comet at any simulation time using the following equation

$$M = \sqrt{\frac{\mu_s}{a^3}}\, t_{pp} = \frac{dM}{dt} t_{pp} = \dot{M}\, t_{pp} = \sqrt{\frac{\mu_s}{a^3}} \left(JD_i - JD_{pp} \right)$$

where μ_s is the gravitational constant of the sun, a is the semimajor axis of the celestial body, and t_{pp} is the time since perihelion passage. In the third form of this equation, JD_i is the simulation Julian day and JD_{pp} is the Julian day of perihelion passage. For consistency, the mean motion rate $\dot{M}$ should be in the units of distance/day since the delta-time $\left(JD_i - JD_{pp} \right)$ is in days.

The semimajor axis of the celestial body is determined from the perihelion distance r_p and orbital eccentricity *e* according to $a = r_p/(1-e)$.

This solution of Kepler's equation in these computer programs is based on a numerical solution devised by Professor J.M.A. Danby at North Carolina State University. Additional information about this algorithm can be found in "The Solution of Kepler's Equation", *Celestial Mechanics*, **31** (1983) 95-107, 317-328 and **40** (1987) 303-312.

The initial guess for Danby's method is $E_0 = M + 0.85\text{sign}(\sin M)e$. The fundamental equation we want to solve is

$$f(E) = E - e\sin E - M = 0$$

which has the first three derivatives given by

$$f'(E) = 1 - e\cos E \qquad f''(E) = e\sin E \qquad f'''(E) = e\cos E$$

The iteration for an updated eccentric anomaly based on a current value E_n is given by the next four equations

$$\Delta(E_n) = -\frac{f}{f'} \qquad \Delta^*(E_n) = -\frac{f}{f' + \frac{1}{2}\Delta f''}$$

$$\Delta_n(E_n) = -\frac{f}{f' + \frac{1}{2}\Delta f'' + \frac{1}{6}\Delta^{*2} f'''} \qquad E_{n+1} = E_n + \Delta_n$$

This algorithm provides quartic convergence of Kepler's equation. This process is repeated until the convergence test $|f(E)| \le \varepsilon$ involving the fundamental equation is satisfied, where ε is the convergence tolerance. This tolerance is *hardwired* in the software to ε =1.0e-10.

Finally, the true anomaly can be calculated with the following two equations

$$\sin\theta = \sqrt{1-e^2}\sin E \qquad \cos\theta = \cos E - e$$

and the four-quadrant inverse tangent given by $\theta = \tan^{-1}(\sin\theta, \cos\theta)$.

If the orbit is hyperbolic, the initial guess is

$$H_0 = \log\left(\frac{2M}{e} + 1.8\right)$$

where H_0 is the hyperbolic anomaly. The fundamental equation and first three derivatives for this case are

$$f(H) = e\sinh H - H - M \qquad f'(H) = e\cosh H - 1$$

$$f''(H) = e\sinh H \qquad f'''(H) = e\cosh H$$

Otherwise, the iteration loop that calculates Δ, Δ^*, and so forth is the same. The true anomaly for hyperbolic orbits is determined with this next set of equations.

$$\sin\theta = \sqrt{e^2 - 1}\sinh H \qquad \cos\theta = e - \cosh H$$

Finally, the true anomaly is determined from a four-quadrant inverse tangent evaluation of these two equations.

The classical orbital elements can be converted to inertial position and velocity vectors or modified equinoctial orbital elements and included in the heliocentric equations of motion.

Here is the Fortran source code that implements this numerical method.

```
      subroutine kepler1 (xma, ecc, eanom, tanom)

c     solve Kepler's equation subroutine

c     Danby's method

c     input

c      xma = mean anomaly (radians)
c      ecc = orbital eccentricity (non-dimensional)

c     output

c      eanom = eccentric anomaly (radians)
c      tanom = true anomaly (radians)

c     Orbital Mechanics Fortran Toolbox

c     *********************************

      implicit double precision (a-h, o-z)

      data pi2 /6.283185307179586d0/

c     define convergence criterion

      data tol /1.0d-10/

      xma = xma - pi2 * int(xma / pi2)

c     initial guess for eccentric anomaly (radians)

      if (ecc .eq. 0.0d0) then

c        circular orbit

         tanom = xma

         eanom = xma

         return

      else if (ecc .lt. 1.0d0) then

c        elliptic orbit

         eanom = xma + 0.85d0 * sign(1.0d0, sin(xma)) * ecc

      else
c        hyperbolic orbit

         eanom = log(2.0d0 * xma / ecc + 1.8d0)

      end if
```

```
      do i = 1, 20
         if (ecc .lt. 1.0d0) then
c           elliptic orbit
            s = ecc * sin(eanom)
            c = ecc * cos(eanom)
            f = eanom - s - xma
            fp = 1.0d0 - c
            fpp = s
            fppp = c
         else
c           hyperbolic orbit
            s = ecc * sinh(eanom)
            c = ecc * cosh(eanom)
            f = ecc * sinh(eanom) - eanom - xma
            fp = c - 1.0d0
            fpp = s
            fppp = c
         end if
c        check for convergence
         if (abs(f) .le. tol) exit
         delta = -f / fp
         delstar = -f / (fp + 0.5d0 * delta * fpp)
         deltak = -f / (fp + 0.5d0 * delstar * fpp
     +             + delstar * delstar * fppp / 6.0d0)
         eanom = eanom + deltak
      enddo
      if (i .gt. 15) then
         pause 'more than 15 iterations in kepler1'
      end if
c     compute true anomaly (radians)
      if (ecc .lt. 1.0d0) then
c        elliptic orbit
         sta = sqrt(1.0d0 - ecc * ecc) * sin(eanom)
         cta = cos(eanom) - ecc
      else
c        hyperbolic orbit
         sta = sqrt(ecc * ecc - 1.0d0) * sinh(eanom)
         cta = ecc - cosh(eanom)
```

```
        end if

        tanom = atan3(sta, cta)

        return
        end
```

SPK ephemerides

Asteroid and comet ephemerides in ASCII or binary SPK format can also be created and retrieved from JPL's *Asteroid & Comet SPK File Generation Request* website located at https://ssd.jpl.nasa.gov/x/spk.html.

Information about SPK format data can be found at https://naif.jpl.nasa.gov/naif/. Finally, be sure to visit the JPL Small-Body Mission-Design Tool located at https://ssd.jpl.nasa.gov/?mdesign_welcome.

"The Solution of Kepler's Equation", *Celestial Mechanics*, **31** (1983) 95-107, 317-328 and **40** (1987) 303-312.

"Procedures For Solving Kepler's Equation", *Celestial Mechanics*, **38** (1986) 307-334.

"A Cubic Approximation For Kepler's Equation", *Celestial Mechanics*, **40** (1987) 329-334.

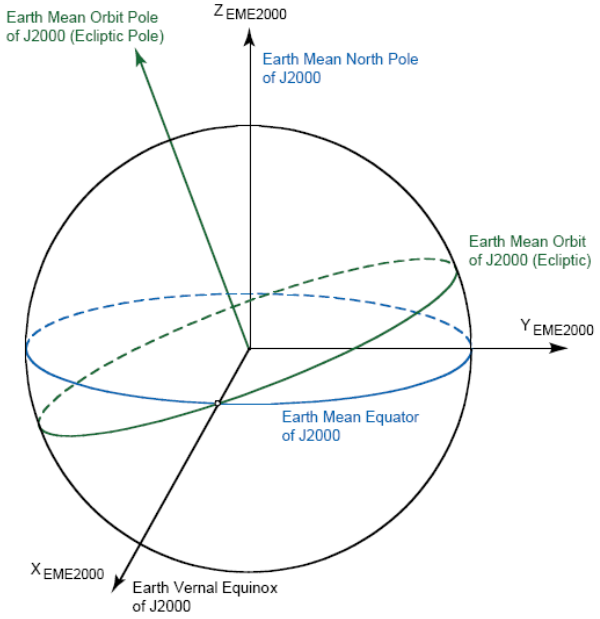
Appendix G

Aerospace Trajectory Coordinates and Time Systems

EME2000 coordinate system

In many of the *CMATO* applications, the spacecraft's orbital motion is modeled with respect to the Earth mean equator and equinox of J2000 (EME2000) coordinate system. The following figure illustrates the geometry of the EME2000 coordinate system. The origin of this Earth-centered-inertial (ECI) coordinate system is the geocenter and the fundamental plane is the Earth's mean equator. The z-axis of this system is normal to the Earth's mean equator at epoch J2000, the x-axis is parallel to the vernal equinox of the Earth's mean orbit at epoch J2000, and the y-axis completes the right-handed coordinate system. The epoch J2000 is the Julian day 2451545.0 which corresponds to January 1, 2000, 12 hours Terrestrial Time (TT).

Terrestrial Time (TT) is the time scale that would be kept by an ideal clock on the geoid - approximately, sea level on the surface of the Earth. Since its unit of time is the SI (atomic) second, TT is independent of the variable rotation of the Earth. TT is meant to be a smooth and continuous "coordinate" time scale independent of Earth rotation. In practice TT is derived from International Atomic Time (TAI), a time scale kept by real clocks on the Earth's surface



Transformation between the J2000 ecliptic and equatorial coordinate systems.

The required matrix-vector transformation is given by

$$\mathbf{V}_{eq} = \begin{bmatrix} 1 & -0.000000479966 & 0 \\ 0.000000440360 & 0.917482137087 & 0.397776982902 \\ -0.000000190919 & -0.397776982902 & 0.917482137087 \end{bmatrix} \mathbf{V}_{ec}$$

where $\mathbf{V}_{ec}$ is a vector in the ecliptic frame, and $\mathbf{V}_{eq}$ is a vector in the equatorial frame. The conversion of equatorial to ecliptic coordinates involves the transpose of this matrix.

Terrestrial Time, TT

Terrestrial Time is the time scale that would be kept by an ideal clock on the geoid - approximately, sea level on the surface of the Earth. Since its unit of time is the SI (atomic) second, TT is independent of the variable rotation of the Earth. TT is meant to be a smooth and continuous "coordinate" time scale independent of Earth rotation. In practice TT is derived from International Atomic Time (TAI), a time scale kept by real clocks on the Earth's surface, by the relation TT = TAI + $32^{s}.184$**.** It is the time scale now used for the precise calculation of future astronomical events observable from Earth.

TT = TAI + 32.184 seconds

TT = UTC + (number of leap seconds) + 32.184 seconds

Barycentric Dynamical Time, TDB

Barycentric Dynamical Time is the time scale that would be kept by an ideal clock, free of gravitational fields, co-moving with the solar system barycenter. It is always within 2 milliseconds of TT, the difference caused by relativistic effects. TDB is the time scale now used for investigations of the dynamics of solar system bodies.

TDB = TT + periodic corrections

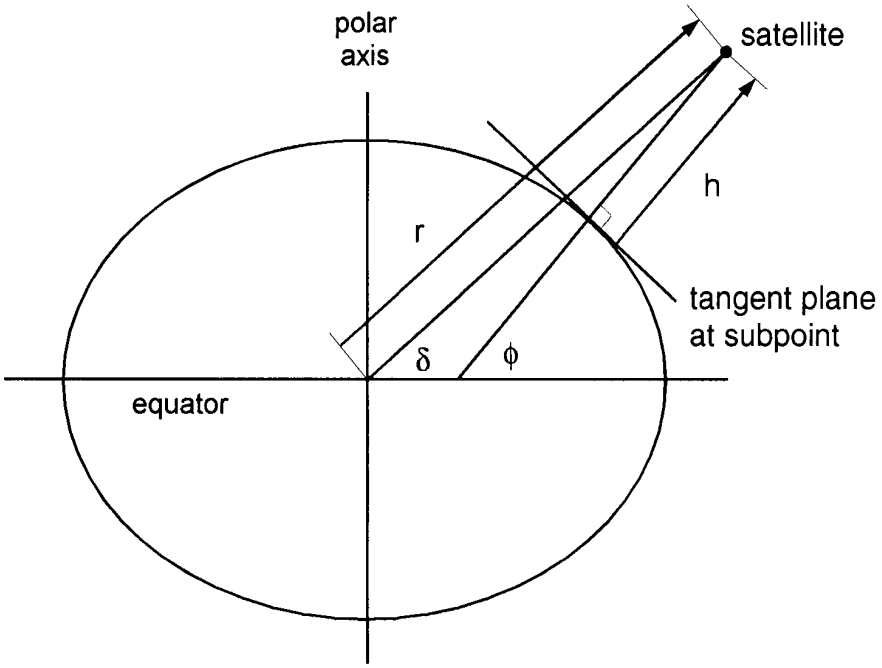
where typical periodic corrections (*USNO Circular 179*) are

$$\begin{aligned} TDB = TT &+ 0.001657\sin\left(628.3076T + 6.2401\right) \\ &+ 0.000022\sin\left(575.3385T + 4.2970\right) \\ &+ 0.000014\sin\left(1256.6152T + 6.1969\right) \\ &+ 0.000005\sin\left(606.9777T + 4.0212\right) \\ &+ 0.000005\sin\left(52.9691T + 0.4444\right) \\ &+ 0.000002\sin\left(21.3299T + 5.5431\right) \\ &+ 0.000010T\sin\left(628.3076T + 4.2490\right) + \cdots \end{aligned}$$

In this equation, the coefficients are in seconds, the angular arguments are in radians, and T is the number of Julian centuries of TT from J2000; T = (Julian day(TT) – 2451545.0) / 36525.

Geodetic coordinates

The following diagram illustrates the geometric relationship between geocentric (radius and declination) and geodetic (altitude and latitude) coordinates.



In this diagram, δ is the geocentric declination, ϕ is the geodetic latitude, *r* is the geocentric distance, and *h* is the geodetic altitude.

The exact mathematical relationship between geocentric and geodetic coordinates is given by the following system of two nonlinear equations

$$(c+h)\cos\phi - r\cos\delta = 0 \qquad (s+h)\sin\phi - r\sin\delta = 0$$

where the geodetic constants *c* and *s* are given by

$$c = \frac{r_{eq}}{\sqrt{1-(2f-f^2)\sin^2\phi}} \qquad s = c(1-f)^2$$

In these equations, r_{eq} is the Earth equatorial radius (6378.14 kilometers) and f is the flattening factor for the Earth (1/298.257).

In these computer programs, the geodetic latitude is determined using the following expression:

$$\phi = \delta + \left(\frac{\sin 2\delta}{\rho}\right) f + \left[\left(\frac{1}{\rho^2} - \frac{1}{4\rho}\right)\sin 4\delta\right] f^2$$

The geodetic altitude is calculated from

$$\hat{h} = (\hat{r}-1) + \left\{\left(\frac{1-\cos 2\delta}{2}\right) f + \left[\left(\frac{1}{4\rho} - \frac{1}{16}\right)(1-\cos 4\delta)\right] f^2\right\}$$

In these equations, ρ is the geocentric distance of the spacecraft, $\hat{h} = h / r_{eq}$ and $\hat{r} = \rho / r_{eq}$.

This algorithm is based on "Derivation of Transformation Formulas Between Geocentric and Geodetic Coordinates for Nonzero Altitudes" by Sheila Ann T. Long, NASA TN D-7522, 1974.

EME2000-to-areocentric coordinate transformation

This section describes the transformation of coordinates between the Earth mean equator and equinox of J2000 (EME2000) and areocentric mean equator and IAU node of epoch coordinate systems. This transformation is used to compute B-plane and other orbital coordinates at encounter.

A unit vector in the direction of the pole of Mars can be determined from

$$\hat{\mathbf{p}}_{Mars} = \begin{bmatrix} \cos\alpha_p \cos\delta_p \\ \sin\alpha_p \cos\delta_p \\ \sin\delta_p \end{bmatrix}$$

The IAU 2000 right ascension and declination of the pole of Mars in the Earth mean equator and equinox of J2000 (EME2000) coordinate system are given by the following expressions

$$\alpha_p = 317.68143 - 0.1061T$$

$$\delta_p = 52.88650 - 0.0609T$$

where T is the time in Julian centuries given by $T = (JD - 2451545.0)/36525$ and JD is the TDB Julian day. The unit vector in the direction of the *IAU-defined* x-axis is computed from $\hat{\mathbf{x}} = \hat{\mathbf{p}}_{J2000} \times \hat{\mathbf{p}}_{Mars}$ where $\hat{\mathbf{p}}_{J2000} = [0\ 0\ 1]^T$ is unit vector in the direction of the north pole of the J2000 coordinate system. The unit vector in the y-axis direction of this coordinate system is $\hat{\mathbf{y}} = \hat{\mathbf{p}}_{Mars} \times \hat{\mathbf{x}}$.

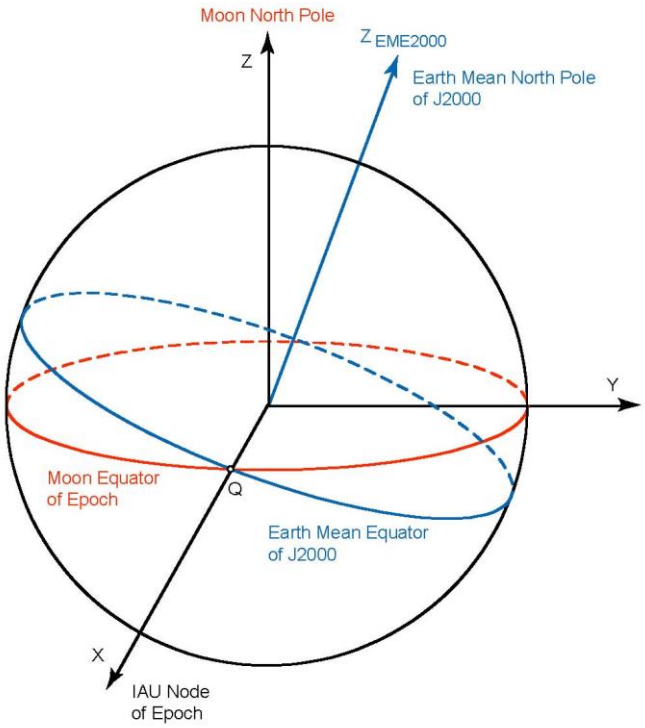
Finally, the components of the matrix that transforms coordinates from the EME2000 system to the Mars-centered mean equator and IAU node of epoch system are as follows

$$\mathbf{M} = \left[\hat{\mathbf{x}} \quad \hat{\mathbf{y}} \quad \hat{\mathbf{p}}_{Mars}\right]^T$$

Geocentric-to-selenocentric coordinate transformation

This section describes the transformation of coordinates between the Earth mean equator and equinox 2000 (EME2000) and lunar mean equator and IAU node of epoch coordinate systems. This transformation is used to compute B-plane coordinates at encounter.

The following diagram illustrates the orientation of the lunar mean equator and IAU node of epoch coordinate frame.



A unit vector in the direction of the north pole of the moon can be determined from

$$\hat{\mathbf{p}}_{Moon} = \begin{bmatrix} \cos\alpha_p \cos\delta_p \\ \sin\alpha_p \cos\delta_p \\ \sin\delta_p \end{bmatrix}$$

where α_p and δ_p are the right ascension and declination of the lunar pole.

The right ascension and declination of the lunar pole in the EME2000 coordinate system are given by the following expressions

$$\begin{aligned} \alpha_P = {} & 269.9949 + 0.0031T - 3.8787\sin E1 - 0.1204\sin E2 \\ & +0.0700\sin E3 - 0.0172\sin E4 + 0.0072\sin E6 \\ & -0.0052\sin E10 + 0.0043\sin E13 \end{aligned}$$

$$\begin{aligned} \delta_P = {} & 66.5392 + 0.0130T + 1.5419\cos E1 + 0.0239\cos E2 \\ & -0.0278\cos E3 + 0.0068\cos E4 - 0.0029\cos E6 \\ & +0.0009\cos E7 + 0.0008\cos E10 - 0.0009\cos E13 \end{aligned}$$

where T is the time in Julian centuries given by $T = \left(JD - 2451545.0\right)/36525$ and JD is the Dynamical Barycentric Time (TDB) Julian Date.

The trigonometric arguments, in degrees, for these equations are

$$E1 = 125.045 - 0.0529921d$$
$$E2 = 250.089 - 0.1059842d$$
$$E3 = 260.008 + 13.0120009d$$
$$E4 = 176.625 + 13.3407154d$$
$$E6 = 311.589 + 26.4057084d$$
$$E7 = 134.963 + 13.0649930d$$
$$E10 = 15.134 - 0.1589763d$$
$$E13 = 25.053 + 12.9590088d$$

where $d = JD - 2451545$ is the number of days since January 1.5, 2000. These equations are given in "Report of the IAU Working Group on Cartographic Coordinates and Rotational Elements: 2009", *Celestial Mechanics and Dynamical Astronomy*, **109**: 101-135, 2011.

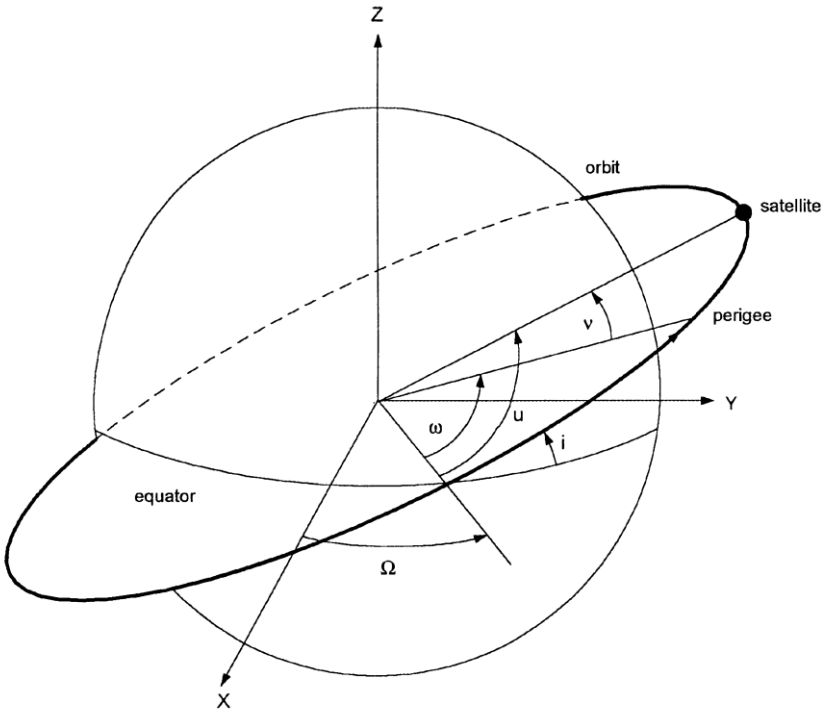
The unit vector in the x-axis direction of this selenocentric coordinate system is given by $\hat{\mathbf{x}} = \hat{\mathbf{z}} \times \hat{\mathbf{p}}_{Moon}$ where $\hat{\mathbf{z}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. The unit vector in the y-axis direction can be determined using $\hat{\mathbf{y}} = \hat{\mathbf{p}}_{Moon} \times \hat{\mathbf{x}}$.

Finally, the components of the matrix that transforms coordinates from the EME2000 system to the moon-centered (selenocentric) mean equator and IAU node of epoch system are

$$\mathbf{M} = \begin{bmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{p}}_{Moon} \end{bmatrix}^T$$

Classical orbital elements

This section describes equations that can be used to convert between inertial state vectors and classical orbital elements. The following diagram illustrates the geometry of these orbital elements.



where

a = semimajor axis

e = orbital eccentricity

i = orbital inclination

ω = argument of perigee

Ω = right ascension of the ascending node

ν = true anomaly

The semimajor axis defines the size of the orbit and the orbital eccentricity defines the shape of the orbit. The angular orbital elements are defined with respect to a fundamental x-axis, the vernal equinox, and a fundamental plane, the equator. The z-axis of this system is collinear with the spin axis of the Earth, and the y-axis completes a right-handed coordinate system.

The orbital inclination is the angle between the equatorial plane and the orbit plane. Spacecraft orbits with inclinations between 0 and 90 degrees are called *direct* orbits and spacecraft with inclinations greater than 90 and less than 180 degrees are called *retrograde* orbits. The right ascension of the ascending node (RAAN) is the angle measured from the x-axis (vernal equinox) eastward along the equator to the ascending node. The argument is the angle from the ascending node, measured along the orbit plane in the direction of increasing true anomaly, to the argument of perigee. The true anomaly is the angle from the argument of perigee, measured along the orbit plane in the direction of motion, to the spacecraft's location. Also shown is the argument of latitude, *u*, which is the angle from the ascending node, measured in the orbit plane, to the spacecraft's location in the orbit. It is equal to $u = \nu + \omega$.

The orbital eccentricity is an indication of the type of orbit. For values of $0 \le e < 1$, the orbit is circular or elliptic. The orbit is parabolic when $e = 1$ and the orbit is hyperbolic if $e > 1$

The semimajor axis is calculated using the following expression:

$$a = \frac{1}{\frac{2}{r} - \frac{v^2}{\mu}}$$

where $r = |\mathbf{r}| = \sqrt{r_x^2 + r_y^2 + r_z^2}$ and $v = |\mathbf{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$. The angular orbital elements are calculated from modified equinoctial orbital elements which are calculated from the ECI position and velocity vectors.

The scalar orbital eccentricity is determined from *h* and *k* as follows:

$$e = \sqrt{h^2 + k^2}$$

The orbital inclination is determined from *p* and *q* using the following expression

$$i = 2\tan^{-1}\left(\sqrt{p^2 + q^2}\right)$$

For values of inclination greater than a small value ε, the right ascension of the ascending node (RAAN) is given by

$$\Omega = \tan^{-1}(p, q)$$

Otherwise, the orbit is equatorial and there is no RAAN. If the value of orbital eccentricity is greater than ε , the argument of perigee is determined from

$$\omega = \tan^{-1}(h, k) - \Omega$$

Otherwise, the orbit is circular and there is no argument of perigee. Typically $\varepsilon = 10^{-8}$.

Finally, the true anomaly is found from the expression $\theta = \lambda - \Omega - \omega$.

In these equations, all two argument inverse tangent calculations use a four quadrant inverse tangent.

The right ascension of a spacecraft is determined from the *x* and *y* components of the ECI *unit* position vector as $\alpha = \tan^{-1}\left(r_{y_{eci}}, r_{x_{eci}}\right)$.

The subpoint of a spacecraft is the point on the Earth's surface directly below the spacecraft. The locus of subpoints describes the ground track of a spacecraft. The east longitude of the subpoint of a spacecraft can be determined from the x and y components of the ECF *unit* position vector as follows:

$$\lambda = \tan^{-1}\left(r_{y_{ecf}}, r_{x_{ecf}}\right)$$

These two position components can be determined from the x and y components of the ECI position vector and the Greenwich sidereal time θ_g as follows:

$$r_{x_{ecf}} = r_{x_{eci}} \cos\theta_g + r_{y_{eci}} \sin\theta_g \qquad r_{y_{ecf}} = r_{y_{eci}} \cos\theta_g - r_{x_{eci}} \sin\theta_g$$

The geocentric declination of the spacecraft subpoint is $\delta = \sin^{-1}(r_z / r)$.

The geocentric declination of perigee and apogee are given by the next two equations.

$$\delta_p = \sin^{-1}(\sin i \sin\omega) \qquad \delta_a = \sin^{-1}\left[\sin i \sin(180 + \omega)\right]$$

If the orbit is elliptic $(a > 0)$, the geocentric radius of perigee and apogee are given by the next two equations.

$$r_p = a(1 - e) \qquad r_a = a(1 + e)$$

The geodetic latitude and altitude of perigee and apogee can be determined from these geocentric coordinates using the geodet1.m conversion function.

The perigee and apogee speed can be determined from

$$V_p = \sqrt{2\mu \frac{r_a}{r_p(r_p + r_a)}} \qquad V_a = \sqrt{2\mu \frac{r_p}{r_a(r_p + r_a)}}$$

If the orbit is hyperbolic $(a < 0)$, there is no apogee and the perigee radius is equal to $r_p = a(1 + e)$

and the perigee speed is equal to

$$V_p = \sqrt{\frac{2\mu}{r_p} + \frac{\mu}{a}}$$

The flight path angle is the angle between the velocity vector and the local horizontal or tangent plane at the spacecraft's location. It can be calculated with the following expression:

$$\gamma = \sin^{-1}\left(\frac{\mathbf{r} \bullet \mathbf{v}}{|\mathbf{r} \bullet \mathbf{v}|}\right)$$

The specific orbital energy, or the energy per unit mass is determined from this next expression:

$$E = \frac{v^2}{2} - \frac{\mu}{r} = -\frac{\mu}{2a}$$

The orbital energy is the sum of a spacecraft's kinetic and potential energy. Notice that it depends only on the spacecraft's semimajor axis and the gravitational constant of the central body.

Hyperbolic orbit

The asymptote unit vector of a hyperbolic orbit is given by

$$\hat{\mathbf{s}} = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

In this expression, α_∞ is the right ascension of the asymptote (RLA), and δ_∞ is the declination of the asymptote (DLA).

The asymptote unit vector at any trajectory time can be computed from

$$\hat{\mathbf{s}} = \frac{1}{1 + C_3 \dfrac{h^2}{\mu^2}} \left\{ \left(\frac{\sqrt{C_3}}{\mu} \right) \mathbf{h} \times \mathbf{e} - \mathbf{e} \right\} = \frac{1}{1 + C_3 \dfrac{p}{\mu}} \left\{ \left(\frac{\sqrt{C_3}}{\mu} \right) \mathbf{h} \times \mathbf{e} - \mathbf{e} \right\}$$

where **h** and **e** are the angular momentum and orbital eccentricity vectors, respectively. In the second expression, *p* is the semiparameter of the hyperbolic orbit which can be computed from

$$p = a\left(1 - e^2\right)$$

The angular momentum and eccentricity vectors are computed using the following equations;

$$\mathbf{h} = \mathbf{r} \times \mathbf{v}$$

$$\mathbf{e} = \frac{\mathbf{v}\times\mathbf{h}}{\mu} - \frac{\mathbf{r}}{r} = \frac{1}{\mu}\left[\left(v^2 - \frac{\mu}{r}\right)\mathbf{r} - (\mathbf{r}\bullet\mathbf{v})\,\mathbf{v}\right]$$

C3 is the "twice specific" orbital energy which is determined from the position **r** and velocity **v** vectors according to

$$C_3 = |\mathbf{v}|^2 - \frac{2\mu}{|\mathbf{r}|}$$

The right ascension and declination of the asymptote can be computed from components of the unit asymptote vector according to

$$\alpha_\infty = \tan^{-1}\left(s_x, s_y\right)$$

$$\delta_\infty = \sin^{-1}\left(s_z\right)$$

The right ascension and declination will be with respect to the coordinate system of the input state vector (EME2000 for example).

Calculating the hyperbolic targets C3, RLA and DLA

Using position and velocity vectors

The asymptote unit vector of a hyperbolic orbit is given by

$$\hat{\mathbf{s}} = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

In this expression, α_∞ is the right ascension of the asymptote (RLA), and δ_∞ is the declination of the asymptote (DLA).

The asymptote unit vector at any trajectory time can be computed from

$$\hat{\mathbf{s}} = \frac{1}{1 + C_3\dfrac{h^2}{\mu^2}}\left\{\left(\frac{\sqrt{C_3}}{\mu}\right)\mathbf{h}\times\mathbf{e} - \mathbf{e}\right\} = \frac{1}{1 + C_3\dfrac{p}{\mu}}\left\{\left(\frac{\sqrt{C_3}}{\mu}\right)\mathbf{h}\times\mathbf{e} - \mathbf{e}\right\}$$

where **h** and **e** are the angular momentum and orbital eccentricity vectors, respectively. In the second expression, *p* is the semi-parameter of the hyperbolic orbit which can be computed from $p = a\left(1 - e^2\right)$.

The angular momentum and eccentricity vectors are computed using the following equations.

$$\mathbf{h} = \mathbf{r}\times\mathbf{v} \qquad \mathbf{e} = \frac{\mathbf{v}\times\mathbf{h}}{\mu} - \frac{\mathbf{r}}{r} = \frac{1}{\mu}\left[\left(v^2 - \frac{\mu}{r}\right)\mathbf{r} - (\mathbf{r}\bullet\mathbf{v})\,\mathbf{v}\right]$$

C3 is the "twice specific" (per unit mass) orbital energy which is determined from the position **r** and velocity **v** vectors according to

$$C_3 = |\mathbf{v}|^2 - \frac{2\mu}{|\mathbf{r}|}$$

The right ascension and declination of the asymptote can be computed from components of the unit asymptote vector according to

$$\alpha_\infty = \tan^{-1}\left(s_x, s_y\right) \qquad \delta_\infty = \sin^{-1}\left(s_z\right)$$

Using classical orbital elements

The asymptote unit vector in terms of the classical orbital elements of a hyperbolic orbit is given by

$$\hat{\mathbf{s}} = \begin{Bmatrix} \cos\Omega\cos\left(\omega+\theta\right) - \sin\Omega\sin\left(\omega+\theta\right)\cos i \\ \sin\Omega\cos\left(\omega+\theta\right) + \cos\Omega\sin\left(\omega+\theta\right)\cos i \\ \sin\left(\omega+\theta\right)\sin i \end{Bmatrix}$$

In this expression, Ω is the right ascension of the ascending node (RAAN), ω is the argument of periapsis and θ is the true anomaly.

The declination of the asymptote (DLA) is given by

$$\delta_\infty = \sin^{-1}\left\{\sin\left(\omega+\theta_\infty\right)\sin i\right\} = \sin^{-1}\left\{\sin\left(u_\infty\right)\sin i\right\}$$

where $u_\infty = \omega + \theta_\infty$ is the argument of latitude of the launch asymptote. In this expression θ_∞ is the true anomaly of the launch hyperbola "at infinity" and is a function of the orbital eccentricity *e* of the hyperbola according to $\theta_\infty = \cos^{-1}\left(-1/e\right)$.

From the following two expressions

$$\sin\left(\alpha_\infty - \Omega\right) = \frac{\tan\delta_\infty}{\tan i} \qquad \cos\left(\alpha_\infty - \Omega\right) = \frac{\cos u_\infty}{\cos\delta_\infty}$$

the right ascension of the asymptote (RLA) can be determined from

$$\alpha_\infty = \Omega + \tan^{-1}\left(\frac{\tan\delta_\infty}{\tan i}, \frac{\cos u_\infty}{\cos\delta_\infty}\right)$$

ECI state vector

The rectangular components of the inertial position vector are determined from

$$r_x = r\left[\cos\Omega\cos\left(\omega+v\right) - \sin\Omega\cos i\sin\left(\omega+v\right)\right]$$

$$r_y = r\left[\sin\Omega\cos\left(\omega+v\right) + \cos\Omega\cos i\sin\left(\omega+v\right)\right]$$

$$r_z = r\sin i\sin\left(\omega+v\right)$$

where $r = \sqrt{r_x^2 + r_y^2 + r_z^2}$ is the scalar distance.

The components of the inertial velocity vector are computed using the following equations:

$$v_x = -\sqrt{\frac{\mu}{p}}\left[\cos\Omega\left\{\sin\left(\omega+v\right)+e\sin\omega\right\}+\sin\Omega\cos i\left\{\cos\left(\omega+v\right)+e\cos\omega\right\}\right]$$

$$v_y = -\sqrt{\frac{\mu}{p}}\left[\sin\Omega\left\{\sin\left(\omega+v\right)+e\sin\omega\right\}-\cos\Omega\cos i\left\{\cos\left(\omega+v\right)+e\cos\omega\right\}\right]$$

$$v_z = \sqrt{\frac{\mu}{p}}\left[\sin i\left\{\cos\left(\omega+v\right)+e\cos\omega\right\}\right]$$

where

a = semimajor axis
e = orbital eccentricity
i = orbital inclination
ω = argument of perigee
Ω = right ascension of the ascending node
v = true anomaly
$p = a\left(1-e^2\right)$

"Planetary Constants and Models", R. Vaughan, JPL D-12947, December 1995.

"Update to Mars Coordinate Frame Definitions", R. A. Mase, JPL IOM 312.B/015-99, 15 July 1999.

"Preliminary Mars Planetary Constants and Models for Mars Sample Return", D. Lyons, JPL IOM 312/99.DTL-1, 20 January 1999.

"Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000", *Celestial Mechanics and Dynamical Astronomy*, **82**: 83-110, 2002.

"Report of the IAU Working Group on Cartographic Coordinates and Rotational Elements: 2009", *Celestial Mechanics and Dynamical Astronomy*, **109**: 101-135, 2011.

"JPL Planetary Ephemeris DE421", E. M. Standish, JPL IOM 312.N-03-009, 24 April 2003.

"IERS Conventions (2003)", IERS Technical Note 32, November 2003.

"Derivation of Transformation Formulas Between Geocentric and Geodetic Coordinates for Nonzero Altitudes" by Sheila Ann T. Long, NASA TN D-7522, 1974.

Appendix H

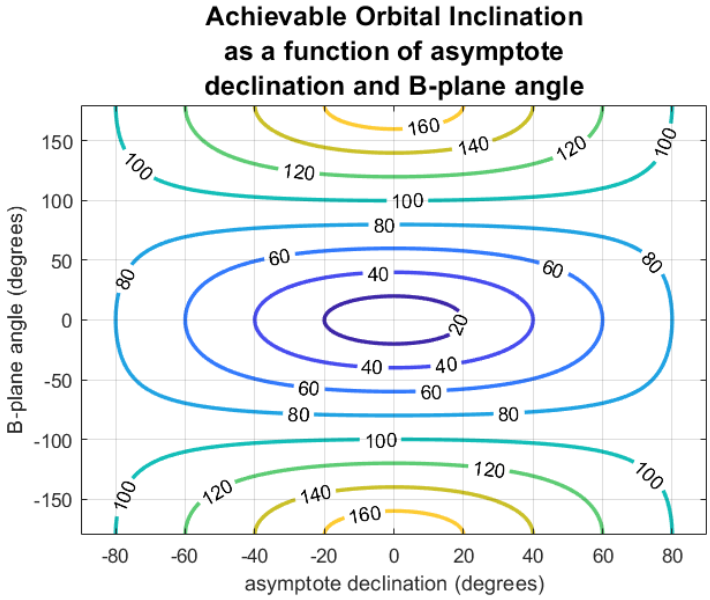
Impulsive Interplanetary Injection from a Circular Earth Orbit

This appendix describes an algorithm that can be used to determine the trajectory characteristics of an interplanetary injection that uses a single impulsive maneuver to transfer a spacecraft from a circular Earth park orbit to a departure hyperbola. The methodology is based on the equations derived in Chapter 4 of Richard Battin's classic text, *Astronautical Guidance*, and Chapter 11 of *An Introduction to the Mathematics and Methods of Astrodynamics*, both written by Professor Battin and published by the American Institute of Aeronautics and Astronautics (AIAA). This algorithm can be used to create a thrust duration initial guess for the `hyper_sos` software.

The numerical algorithm is valid for geocentric orbit inclinations that satisfy the following geometric constraint $i \geq |\delta_\infty|$ where *i* is the orbital inclination of the park orbit. The algorithm also assumes a circular Earth park orbit and that injection occurs impulsively at the perigee of the departure hyperbola.

Whenever $i > |\delta_\infty|$, there will be two opportunities to establish a departure hyperbola that will satisfy the energy and orientation of the outgoing asymptote. Typically, one injection opportunity will occur when the spacecraft is ascending and the other when the spacecraft is descending along the park orbit. For the case where $i = |\delta_\infty|$, there will be a single injection opportunity.

The relationship between orbital inclination *i*, B-plane angle θ and asymptote declination δ of an incoming or outgoing hyperbola is given by $\cos i = \cos\theta \cos\delta$. The following is a contour plot illustrating the achievable inclination as a function of B-plane angle and declination.



Coplanar transfer – orientation of the park orbit and departure hyperbola

This section summarizes the equations used to determine the right ascension of the ascending node (RAAN) of the park orbit and the injection true anomaly on the park orbit.

A unit vector is the direction of the departure asymptote is given by

$$\hat{\mathbf{s}} = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

where

$\alpha_\infty =$ right ascension of departure asymptote

$\delta_\infty =$ declination of departure asymptote

The angle between the outgoing asymptote and the spin axis of the Earth is given by $\beta = \cos^{-1}\left(\hat{\mathbf{s}} \bullet \hat{\mathbf{z}}\right)$ where $\hat{\mathbf{z}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. Note that $\beta = 90^\circ - \delta_\infty$.

The park orbit right ascension of the ascending node for each opportunity can be determined from

$$\Omega_1 = 180^\circ + \alpha_\infty + \sin^{-1}\left(\frac{\cot\beta}{\tan i}\right) \qquad \Omega_2 = 360^\circ + \alpha_\infty - \sin^{-1}\left(\frac{\cot\beta}{\tan i}\right)$$

The true anomaly on the park orbit for each injection opportunity can be determined from

$$\theta_1 = \cos^{-1}\left(\frac{\cos\beta}{\sin i}\right) - \eta \qquad \theta_2 = -\cos^{-1}\left(\frac{\cos\beta}{\sin i}\right) - \eta$$

where

$$\eta = \sin^{-1}\left(\frac{1}{1 + r_p V_\infty^2 / \mu}\right)$$

In the last equation, r_p is the geocentric radius of the park orbit and μ is the gravitational constant of the Earth. The speed of a spacecraft "at infinity" is determined from $V_\infty = \sqrt{C_3}$.

For a tangential impulsive injection maneuver that occurs at perigee of the hyperbola, the true anomaly on the hyperbola orbit is zero. Furthermore, since the orbit transfer modeled for this case is coplanar, the right ascension of the ascending node (RAAN) computed above should be the same for both the park orbit and the launch hyperbola. This can be verified by examining the hyperbola's RAAN which is computed using the state vector at injection.

Coplanar transfer – departure delta-v

The velocity vector at any geocentric position vector $\mathbf{r}$ required to achieve a launch hyperbola defined by V_∞, α_∞ and δ_∞ is given by

$$\mathbf{V}_h = \left(d + \frac{1}{2}V_\infty\right)\hat{\mathbf{s}} + \left(d - \frac{1}{2}V_\infty\right)\hat{\mathbf{r}}$$

where

$$d = \sqrt{\frac{\mu}{(1+\cos\psi)r_p} + \frac{V_\infty^2}{4}}$$

and ψ is the angle between the spacecraft's position vector and the departure asymptote unit vector which can be computed using $\cos\psi = \hat{\mathbf{s}} \bullet \hat{\mathbf{r}}$.

The injection $\Delta\mathbf{v}$ vector can be determined from $\Delta\mathbf{V} = \mathbf{V}_h - \mathbf{V}_p$ where $\mathbf{v}_p$ is the inertial velocity vector in the park orbit at the impulsive injection and $\hat{\mathbf{r}} = \mathbf{r}/|\mathbf{r}|$.

Finally, the scalar injection delta-v is $\Delta V = |\Delta\mathbf{V}|$.

Non-coplanar transfer – park orbit orientation and departure delta-V

A geocentric unit vector is the direction of the departure asymptote is given by

$$\hat{\mathbf{i}}_\infty = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

where

$\alpha_\infty =$ right ascension of departure asymptote

$\delta_\infty =$ declination of departure asymptote

The velocity vector of the spacecraft on the initial circular orbit is given by $\mathbf{V}_0 = \sqrt{\mu/r}\,\hat{\mathbf{i}}_\theta$.

The velocity vector at any geocentric position vector $\mathbf{r}$ required to achieve a departure hyperbola defined by v_∞, α_∞ and δ_∞ is given by $\mathbf{V}_1 = \frac{1}{2}V_\infty\left[(D+1)\hat{\mathbf{i}}_\infty + (D-1)\hat{\mathbf{i}}_r\right]$

where

$$D = \sqrt{1 + \frac{4\mu}{rV_\infty^2\left(1+\hat{\mathbf{i}}_\infty \bullet \hat{\mathbf{i}}_r\right)}}$$

and

$$\hat{\mathbf{i}}_r = \begin{bmatrix} \cos\Omega\cos\theta - \sin\Omega\sin\theta\cos i \\ \sin\Omega\cos\theta + \cos\Omega\sin\theta\cos i \\ \sin\theta\sin i \end{bmatrix} \qquad \hat{\mathbf{i}}_\theta = \begin{bmatrix} -\cos\Omega\sin\theta - \sin\Omega\cos\theta\cos i \\ -\sin\Omega\sin\theta + \cos\Omega\cos\theta\cos i \\ \cos\theta\sin i \end{bmatrix}$$

In these equations, Ω is the right ascension of the ascending node, i is the orbital inclination, θ is the true anomaly at injection, r is the geocentric radius of the park orbit and $V_\infty = \sqrt{C_3}$.

The injection $\Delta\mathbf{V}$ vector can be determined from $\Delta\mathbf{V} = \mathbf{V}_1 - \mathbf{V}_0$. Finally, the scalar injection delta-v is $\Delta V = |\Delta\mathbf{V}|$.

The orientation of the park orbit and departure hyperbola at injection is computed using a two-dimensional grid search involving the park orbit right ascension of the ascending node (RAAN) and the true anomaly of the impulsive maneuver on the park orbit. During the grid search, the software uses a nonlinear programming (NLP) algorithm which is part of *Sparse Optimization Suite* to find the current minimum delta-v and saves the RAAN and true anomaly values corresponding to the "best" delta-v.

Example calculation

Here are the park orbit and departure hyperbola characteristics for a typical example.

- park orbit altitude = `185.2` kilometers
- park orbit inclination = `28.5` degrees
- C3 of the departure hyperbola = `9.28` km^2/sec^2
- right ascension of outgoing asymptote = `352.59` degrees
- declination of the outgoing asymptote = `2.27` degrees

The following is the numerical summary for this example. We can see that there are two injection opportunities with different RAANs and injection true anomalies. However, the scalar magnitude of the injection delta-v is identical.

```
-------------------------------------------------
Interplanetary Injection from a Circular Park Orbit
-------------------------------------------------

departure hyperbola characteristics
-----------------------------------

c3                          9.28000000000000      km**2/sec**2

asymptote right ascension   352.590000000000      degrees
asymptote declination       2.27000000000000      degrees

orbital elements and state vector of park orbit at injection - opportunity #1
------------------------------------------------------------------------------

     sma (km)          eccentricity       inclination (deg)     argper (deg)
 0.6563340000D+04    0.0000000000D+00    0.2850000000D+02    0.0000000000D+00

    raan (deg)      true anomaly (deg)     arglat (deg)        period (min)
 0.1767767337D+03    0.2507477991D+02    0.2507477991D+02    0.8819562703D+02

     rx (km)             ry (km)             rz (km)             rmag (km)
 -.6072821513D+04    -.2106348521D+04    0.1327240269D+04    0.6563340000D+04

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 0.2948681176D+01    -.6379088912D+01    0.3368063861D+01    0.7793032157D+01

orbital elements and state vector of hyperbola at injection - opportunity #1
-----------------------------------------------------------------------------

     sma (km)          eccentricity       inclination (deg)     argper (deg)
 -.4295264009D+05    0.1152804111D+01    0.2850000000D+02    0.2507477991D+02

    raan (deg)      true anomaly (deg)     arglat (deg)
 0.1767767337D+03    0.3600000000D+03    0.2507477991D+02
```

```
     rx (km)             ry (km)             rz (km)             rmag (km)
 -.6072821513D+04    -.2106348521D+04    0.1327240269D+04    0.6563340000D+04

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 0.4326433915D+01    -.9359678097D+01    0.4941770522D+01    0.1143427743D+02
```

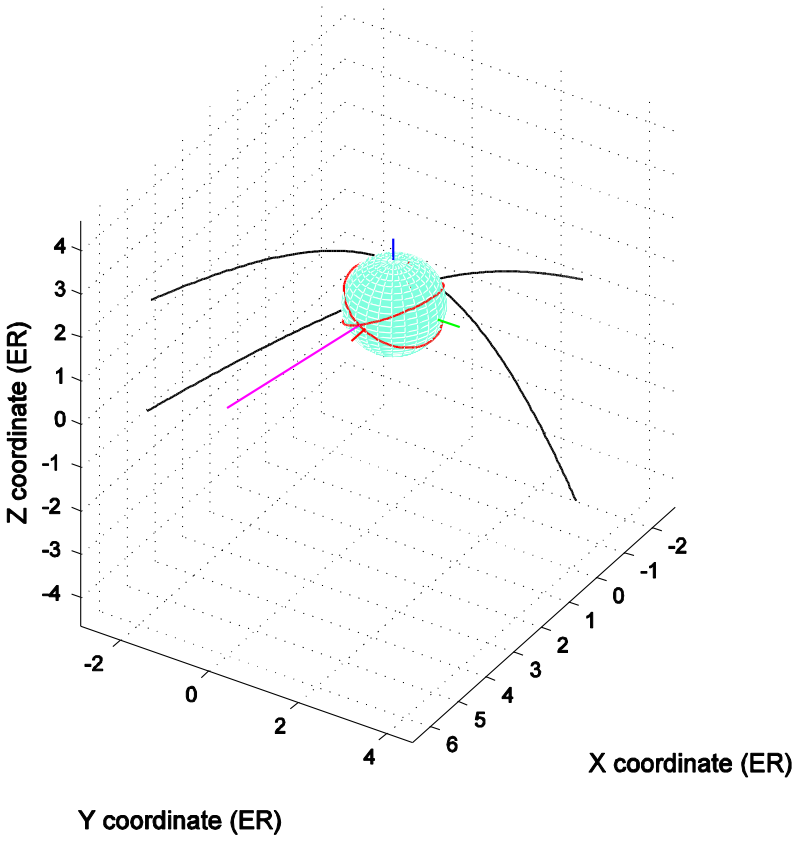
injection delta-v vector and magnitude - opportunity #1

```
x-component of delta-v       1377.75273908206      meters/second

y-component of delta-v      -2980.58918468226      meters/second

z-component of delta-v       1573.70666162370      meters/second

delta-v magnitude            3641.24527527765      meters/second

orbital elements and state vector of park orbit at injection - opportunity #2
-------------------------------------------------------------------------------

     sma (km)          eccentricity       inclination (deg)     argper (deg)
 0.6563340000D+04    0.0000000000D+00    0.2850000000D+02    0.0000000000D+00

    raan (deg)      true anomaly (deg)     arglat (deg)        period (min)
 0.3484032663D+03    0.2145979019D+03    0.2145979019D+03    0.8819562703D+02

     rx (km)             ry (km)             rz (km)             rmag (km)
 -.5950748689D+04    -.2122224678D+04    -.1778253190D+04    0.6563340000D+04

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 0.3201396036D+01    -.6411955694D+01    -.3060921069D+01    0.7793032157D+01

orbital elements and state vector of hyperbola at injection - opportunity #2
-----------------------------------------------------------------------------

     sma (km)          eccentricity       inclination (deg)     argper (deg)
 -.4295264009D+05    0.1152804111D+01    0.2850000000D+02    0.2145979019D+03

    raan (deg)      true anomaly (deg)     arglat (deg)
 0.3484032663D+03    0.3600000000D+03    0.2145979019D+03

     rx (km)             ry (km)             rz (km)             rmag (km)
 -.5950748689D+04    -.2122224678D+04    -.1778253190D+04    0.6563340000D+04

     vx (kps)            vy (kps)            vz (kps)            vmag (kps)
 0.4697228205D+01    -.9407901676D+01    -.4491117193D+01    0.1143427743D+02
```

injection delta-v vector and magnitude - opportunity #2

```
x-component of delta-v       1495.83216868269      meters/second

y-component of delta-v      -2995.94598183915      meters/second

z-component of delta-v      -1430.19612353262      meters/second

delta-v magnitude            3641.24527527765      meters/second
```

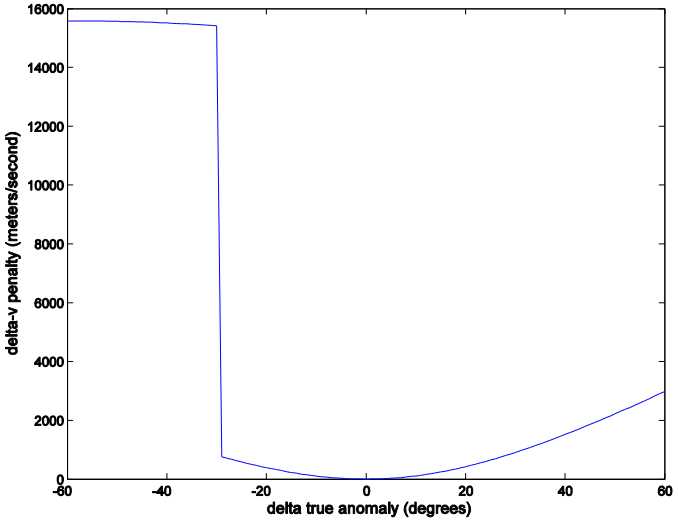
The following is a graphics display illustrating the two injection opportunities for this example. This display is labeled with an Earth centered, inertial (ECI) coordinate system. The x-axis of this system is red, the y-axis is green and the z-axis is blue. The outgoing asymptote is colored magenta, the park orbit traces are red, and the hyperbolic trajectories are black. Please note the units for each coordinate axis are Earth radii (ER).



Delta-v penalty for off-nominal injection

The velocity-required equation given above can also be used to access the delta-v penalty for off-nominal injection. Such things as ignition timing errors and other spacecraft contingencies may result in an injection maneuver that does not occur at the optimal true anomaly on the park orbit.

The following plot illustrates how the injection delta-v penalty changes as the true anomaly at injection is displaced from the optimal.



"Design of Lunar and Interplanetary Ascent Trajectories", Victor C. Clarke, Jr., JPL Technical Report No. 32-30, March 15, 1962.

"Optimal Noncoplanar Escape from Circular Orbits", T. N. Edelbaum, *AIAA Journal*, Vol. 9, No. 12, December 1971.

"Interplanetary Mission Design Handbook, Volume 1, Part 2", JPL Publication 82-43, September 15, 1983.

An Introduction to the Mathematics and Methods of Astrodynamics, Richard H. Battin, AIAA Education Series, 1987.

Appendix I

Numerical Solutions of Lambert's Problem

Lambert's problem is concerned with the determination of a two-body orbit that passes between two positions within a specified time-of-flight. This classic astrodynamic problem is also known as the orbital two-point boundary value problem (TPBVP).

The time to traverse a trajectory depends only upon the length of the semimajor axis *a* of the transfer trajectory, the sum $r_i + r_f$ of the distances of the initial and final positions relative to a central body, and the length *c* of the chord joining these two positions. This relationship can be stated functionally as $tof = tof\left(r_i + r_f, c, a\right)$.

From the following form of Kepler's equation

$$t - t_0 = \sqrt{\frac{a^3}{\mu}\left(E - e\sin E\right)}$$

we can write

$$t = \sqrt{\frac{a^3}{\mu}}\left[E - E_0 - e\left(\sin E - \sin E_0\right)\right]$$

where *E* is the eccentric anomaly associated with radius *r*, E_0 is the eccentric anomaly at r_0, and $t = 0$ when $r = r_0$.

At this point we need to introduce the following trigonometric sun and difference identities:

$$\sin\alpha - \sin\beta = 2\sin\frac{\alpha - \beta}{2}\cos\frac{a + \beta}{2}$$

$$\cos\alpha - \cos\beta = -2\sin\frac{\alpha - \beta}{2}\sin\frac{a + \beta}{2}$$

$$\cos\alpha + \cos\beta = 2\cos\frac{\alpha - \beta}{2}\cos\frac{a + \beta}{2}$$

If we let $E = \alpha$ and $E_0 = \beta$, and substitute the first trig identity into the second equation above, we have the following equation

$$t = \sqrt{\frac{a^3}{\mu}}\left\{E - E_0 - 2\sin\frac{E - E_0}{2}\left(e\cos\frac{E + E_0}{2}\right)\right\}$$

With the two substitutions given by

$$e\cos\frac{E + E_0}{2} = \cos\frac{\alpha + \beta}{2} \qquad \sin\frac{E - E_0}{2} = \sin\frac{\alpha - \beta}{2}$$

the time equation becomes

$$t = \sqrt{\frac{a^3}{\mu}\left\{(\alpha - \beta) - 2\sin\frac{\alpha - \beta}{2}\cos\frac{\alpha + \beta}{2}\right\}}$$

From the elliptic relationships given by

$$r = a(1 - e\cos E)$$

$$x = a(\cos E - e)$$

$$y = a\sin E\sqrt{1 - e^2}$$

and some more manipulation, we have the following equations

$$\cos\alpha = \left(1 - \frac{r + r_0}{2a}\right) - \frac{c}{2a} = 1 - \frac{r + r_0 + c}{2a} = 1 - \frac{s}{a}$$

$$\sin\beta = \left(1 - \frac{r + r_0}{2a}\right) + \frac{c}{2a} = 1 - \frac{r + r_0 - c}{2a} = 1 - \frac{s - c}{a}$$

This part of the derivation makes use of the following three relationships

$$\cos\frac{\alpha - \beta}{2}\cos\frac{\alpha + \beta}{2} = 1 - \frac{r + r_0}{2}$$

$$\sin\frac{\alpha - \beta}{2}\sin\frac{\alpha + \beta}{2} = \sin\frac{E - E_0}{2}\sqrt{1 - \left(e\cos\frac{E + E_0}{2}\right)^2}$$

$$\left(\sin\frac{\alpha - \beta}{2}\sin\frac{\alpha + \beta}{2}\right)^2 = \left(\frac{x - x_0}{2a}\right)^2 + \left(\frac{y - y_0}{2a}\right)^2 = \left(\frac{c}{2a}\right)^2$$

With the use of the half angle formulas given by

$$\sin\frac{\alpha}{2} = \sqrt{\frac{s}{2a}} \qquad \sin\frac{\beta}{2} = \sqrt{\frac{s - c}{2a}}$$

and several additional substitutions, we have the time-of-flight form of Lambert's theorem

$$t = \sqrt{\frac{a^3}{\mu}}\left[(\alpha - \beta) - (\sin\alpha - \sin\beta)\right]$$

A discussion about the angles α and β can be found in "Geometrical Interpretation of the Angles α and β in Lambert's Problem" by J. E. Prussing, AIAA *Journal of Guidance and Control*, Volume 2, Number 5, Sept.-Oct. 1979, pages 442-443.

Gooding's solution of Lambert's problem

The algorithm used in the *Computer Methods for Aerospace Trajectory Optimization* software suite is based on the method described in "A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem" by R. H. Gooding, *Celestial Mechanics and Dynamical Astronomy* **48:** 145-165, 1990. This iterative solution is valid for elliptic, parabolic and hyperbolic transfer orbits which may be either posigrade or retrograde, and involve one or more revolutions about the central body.

Gedeon's solution of Lambert's problem

Another practical numerical method for solving Lambert's problem is described in "A Practical Note on the Use of Lambert's Equation" by Geza Gedeon, *AIAA Journal*, Volume 3, Number 1, 1965, pages 149-150. This iterative solution is valid for elliptic, parabolic and hyperbolic transfer orbits which may be either posigrade or retrograde, and involve one or more revolutions about the central body. Additional information can also be found in G. S. Gedeon, "Lambertian Mechanics", Proceedings of the 12th International Astronautical Congress, Vol. I, 172-190.

The *elliptic* form of the general Lambert Theorem is

$$t = \sqrt{\frac{a^3}{\mu}}\left[(1-k)m\pi + k(\alpha - \sin\alpha) \mp (\beta - \sin\beta)\right]$$

where k may be either +1 (posigrade) or –1 (retrograde), and m is the number of revolutions about the central body.

The Gedeon algorithm introduces the following parameter

$$z = \frac{s}{2a}$$

and solves the problem with a Newton-Raphson procedure. In this equation, a is the semimajor axis of the transfer orbit and

$$s = \frac{r_1 + r_2 + c}{2}$$

This algorithm also makes use of the following constant

$$w = \pm\sqrt{1 - \frac{c}{s}}$$

The function to be solved iteratively is given by:

$$N(z) = \frac{1}{z|z|^{1/2}\,2^{1/2}}\left\{\frac{1-k}{2}m\pi + k\left[|z|^{1/2} - |z|^{1/2}(1-z)^{1/2}\right] - \left[w|z|^{1/2} - w|z|^{1/2} - w|z|^{1/2}\left(1-w^2z\right)^{1/2}\right]\right\}$$

The Newton-Raphson algorithm also requires the derivative of this equation given by

$$N'(z) = \frac{dN}{dz} = \frac{1}{|z|2^{1/2}} \left\{ \frac{k}{(1-z)^{1/2}} - \frac{w^3}{(1-w^2 z)^{1/2}} - \frac{3N(z)}{2^{1/2}} \right\}$$

The iteration for *z* is as follows

$$z_{n+1} = z_n - \frac{N(z_n)}{N'(z_n)}$$

Perturbed motion solutions of Lambert's problem

Shooting method with state transition matrix updates

An initial guess for this algorithm is created by first solving the two-body form of Lambert's problem. At each *shooting* iteration, the initial delta-velocity vector is updated according to

$$\Delta \mathbf{V} = \left[\Phi_{12}\right]^{-1} \Delta \mathbf{r}$$

where the error in the final position vector $\Delta \mathbf{r}$ is determined from the difference between the two body final position vector $\mathbf{r}_{tb}$ and the final position vector predicted by numerical integration $\mathbf{r}_{\text{int}}$ of the orbital equations of motion as follows

$$\Delta \mathbf{r} = \mathbf{r}_{tb} - \mathbf{r}_{\text{int}}$$

The new initial velocity vector can now be calculated from

$$\mathbf{V}_{n+1} = \mathbf{V}_n + \Delta \mathbf{V}$$

The sub-matrix Φ_{12} of the full state transition matrix is as follows:

$$\Phi_{12} = \left[\frac{\partial \mathbf{r}}{\partial \mathbf{V}_0} \right] = \begin{bmatrix} \partial x / \partial \dot{x}_0 & \partial x / \partial \dot{y}_0 & \partial x / \partial \dot{z}_0 \\ \partial y / \partial \dot{x}_0 & \partial y / \partial \dot{y}_0 & \partial y / \partial \dot{z}_0 \\ \partial z / \partial \dot{x}_0 & \partial z / \partial \dot{y}_0 & \partial z / \partial \dot{z}_0 \end{bmatrix}$$

This sub-matrix consists of the partial derivatives of the rectangular cartesian components of the final position vector with respect to the initial velocity vector.

Nonlinear programming solution

In this classic trajectory optimization problem, the components of the initial and final delta-v vectors are the *control variables* and the scalar magnitude of the flyby or rendezvous ΔV is the *objective function* or *performance index*. The NLP implementation uses the two-body solution for Lambert's problem as its initial guess.

For the flyby problem, this method attempts to match all three components of the position vector. For the rendezvous problem, the NLP attempts to match all three components of both the target position and velocity vectors. These mission requirements are formulated as equality constraints.

"A Practical Note on the Use of Lambert's Equation" Geza Gedeon, *AIAA Journal*, Volume 3, Number 1, 1965, pages 149-150.

An Introduction to the Mathematics and Methods of Astrodynamics, Richard H. Battin, AIAA Education Series, 1987.

Analytical Mechanics of Space Systems, Hanspeter Schaub and John L. Junkins, AIAA Education Series, 2003.

Orbital Mechanics, Vladimir A. Chobotov, AIAA Education Series, 2002.

Modern Astrodynamics, Victor R. Bond and Mark C. Allman, Princeton University Press, 1996.

Spacecraft Mission Design, Charles D. Brown, AIAA Education Series, 1992.

Appendix J

Sparse Optimization Suite Configuration File

This suite of trajectory optimization computer programs can read and use a user-defined *SOS* configuration file. A description of each element in this file can be found in the **INSOCX** routine described in section 6.2, *Subprograms for Optimal Control*, and the **INSNLP** routine in Section 2.2, *Subprograms for Optimization* both documented in the *Sparse Optimization Suite* user's manual.

Please note the software can read and use a subset of the information in this file. For example, a subset configuration file might contain only the following information;

```
ODETOL=0.1D-06
INSNLP:IOFLAG=5
SOCOUT=I4K4
```

The following is a typical "full version" configuration file created during the execution of the `oneburn_sos` software.

```
AEQTOL=0.1000000000000000D-02
DTAUX=0.0000000000000000D+00
OBJCTL=0.1000000000000000D-04
ODETOL=0.1000000011686097D-06
PGDCTL=0.1000000000000000D-02
PRTMSD=0.1490116119384766D-07
PRTMXD=0.1000000000000000D-02
PRTSFD=0.1000000000000000D-04
QDRTOL=0.1000000000000000D-02
RESTOL=0.1000000000000000D-04
SMLTOL=0.1490116119384766D-10
TOLJSD=0.1000000000000000D-05
TOLM5A=0.1490116119384766D-07
TOLM5R=0.1490116119384766D-07
IDSCPH=0
IDSCND=0
IDSCVR=0
IDSCFN=0
IDTSFD=-1
IPFAUX=0
IPFSFD=0
IPRSFD=1
IPGRD=0
IPNLP=10
IPODE=0
IPUAUX=0
IPUOCP=6
IRSTRT=2
ISCALE=0
ISFHES=41
ISFINP=42
ISFRST=43
ISFSCL=44
ITSWCH=2
M5DTYP=0
MITODE=20
MTSWCH=-1
MXDATA=0
MXPARM=10
MXPCON=20
MXSTAT=20
MXTERM=50
NPTAUX=100
NSSWCH=-1
SOCOUT=A0B0C0D0E0F0G0H0I0J2K0L0M0N0O0P0Q0R0S1T0U0V0W0X0Y0Z0
SPRTHS=SPARSE
NLPALG=SNLPMN
```

```
NLPOMR=M
KEYDPL=.lueiLUE
RHSTMP=RHSTMPLT
RSTFIL=tlto1.rsbin
SCLFIL=scalewgt.fil
INSNLP:ALFLWR=0.0000000000000000D+00
INSNLP:ALFUPR=0.1000000000000000D+01
INSNLP:CONTOL=0.1490116119384766D-07
INSNLP:EPSRLF=0.1490116119384766D-07
INSNLP:OBJTOL=0.9999999747378752D-05
INSNLP:PGDTOL=0.1000000000000000D-04
INSNLP:SLPTOL=0.9000000000000000D+00
INSNLP:SFZTOL=0.1000000000000000D-01
INSNLP:TOLFIL=0.2000000000000000D+01
INSNLP:TOLKTC=0.1110953834938985D+26
INSNLP:TOLPVT=0.1000000000000000D-02
INSNLP:IHESHN=0
INSNLP:IOFLAG=5
INSNLP:IOFLIN=-1
INSNLP:IOFMFR=0
INSNLP:IOFPAT=0
INSNLP:IOFSHR=0
INSNLP:IOFSRC=0
INSNLP:IPUDRF=0
INSNLP:IPUFZF=0
INSNLP:IPUMF1=11
INSNLP:IPUMF2=12
INSNLP:IPUMF3=13
INSNLP:IPUMF4=14
INSNLP:IPUMF5=15
INSNLP:IPUMF6=16
INSNLP:IPUMF7=17
INSNLP:IPUNLP=6
INSNLP:IPUSTF=0
INSNLP:IRELAX=1
INSNLP:ITDRQP=-1
INSNLP:ITFZQP=-1
INSNLP:IT1MAX=20
INSNLP:JACPRM=0
INSNLP:LYNFNC=0
INSNLP:LYNOUT=0
INSNLP:LYNPLT=0
INSNLP:LYNPNT=101
INSNLP:LYNVAR=0
INSNLP:MAXLYN=5
INSNLP:MAXNFE=50000
INSNLP:MNSAME=2
INSNLP:NEWTON=0
INSNLP:NITMAX=1000
INSNLP:NITMIN=0
INSNLP:NORMAL=0
INSNLP:ALGOPT=FM
INSNLP:KTOPTN=SMALL
INSNLP:QPOPTN=SPARSE
INSNLP:BIGCON=-0.1000000000000000D+01
INSNLP:FEATOL=0.1000000000000000D-01
INSNLP:PMULWR=0.1000000000000000D+00
INSNLP:PTHTOL=0.1000000000000000D+02
INSNLP:RHOLWR=0.1000000000000000D+03
INSNLP:IMAXMU=10
INSNLP:MUCALC=3
INSNLP:MXQPIT=1
```

Appendix K

Example Fortran Subroutines

This appendix presents several typical Fortran subroutines used in the software implementation of *CMATO* applications. Each subroutine is annotated to help explain the processing.

This first example illustrates a typical subroutine that calculates an initial guess.

```
      subroutine odeigs (iphase, ngrdpt, t, z, nz, iferr)

c     user-defined initial guess subroutine

c     **********************************

      implicit double precision (a-h, o-z)

      save

      include 'iofiles.inc'

      include 'socscom1.inc'

      dimension rp1(3), vp1(3), rp2(3), vp2(3), rp3(3), vp3(3)

      dimension rti1(3), vti1(3), rtf1(3), vtf1(3)

      dimension rti2(3), vti2(3), rtf2(3), vtf2(3)

      dimension rti3(3), vti3(3), rtf3(3), vtf3(3)

      dimension r1(3), v1(3), r2(3), v2(3), r3(3), v3(3)

      dimension z(6), statev(12, 11), vinf_in(3), vinf_out(3)

      iferr = 0

c     current julian dates

      xjdate1 = xjdatei1

      xjdate2 = xjdatei1 + td1

      xjdate3 = xjdatei1 + td1 + td2

c     compute heliocentric state vector of departure planet (km & km/sec)

      call p2000(ip1, xjdate1, rp1, vp1)

c     compute heliocentric state vector of flyby planet (km & km/sec)

      call p2000(ip2, xjdate2, rp2, vp2)

c     compute heliocentric state vector of arrival planet (km & km/sec)

      call p2000(ip3, xjdate3, rp3, vp3)

c     -------------------------------------
c     solve lambert's problem for first leg
c     -------------------------------------

c     posigrade transfer
```

```
      direct = 1.0d0

c     less than one rev transfer

      revmax = 0.0d0

c     process first (and only) solution

      isn = 1

      tof1 = 86400.0d0 * (xjdate2 - xjdate1)

      call lambfunc(smu, rp1, rp2, tof1, direct, revmax,
     &              statev, nsol)

      if (nsol .eq. 0) then

         pause 'nsol = 1'

         iferr = -1

         return

      end if

c     initial position vector of first heliocentric leg (km)

      do i = 1, 3

         rti1(i) = rp1(i)

      end do

c     extract initial velocity vector of first heliocentric leg (km/sec)

      vti1(1) = statev(4, isn)
      vti1(2) = statev(5, isn)
      vti1(3) = statev(6, isn)

c     propagate transfer orbit from launch to flyby

      tol = 1.0d-10

      tof = 86400.0d0 * (xjdate2 - xjdate1)

      call twobody2 (smu, tol, tof, rti1, vti1, rtf1, vtf1)

c     compute incoming v-infinity vector and magnitude

      do i = 1, 3

         vinf_in(i) = vtf1(i) - vp2(i)

      end do

      vinfm_in = vecmag(vinf_in)

c     ********************
c     flyby-to-arrival leg
c     ********************

c     initial heliocentric position vector of the transfer trajectory

      do i = 1, 3

         rti2(i) = rp2(i)
```

```
      end do

c     solve lambert's problem for second leg of transfer

      isn = 1

      direct = 1.0d0

      revmax = 0.0d0

      tof = 86400.0d0 * (xjdate3 - xjdate2)

      call lambfunc(smu, rp2, rp3, tof, direct, revmax,
     &              statev, nsol)

c     velocity vector of heliocentric transfer trajectory at flyby

      vti2(1) = statev(4, isn)
      vti2(2) = statev(5, isn)
      vti2(3) = statev(6, isn)

c     compute outgoing v-infinity vector and magnitude

      do i = 1, 3

         vinf_out(i) = vti2(i) - vp2(i)

      end do

      vinfm_out = vecmag(vinf_out)

c     propagate transfer orbit from flyby to arrival

      tol = 1.0d-10

      tof = 86400.0d0 * (xjdate3 - xjdate2)

      call twobody2 (smu, tol, tof, rti2, vti2, rtf2, vtf2)

      if (iphase .eq. 1) then

c        ****************************************************************
c        create state vector on first leg of transfer orbit at grid points
c        ****************************************************************

         deltat = (xjdate2 - xjdate1) / (ngrid1 - 1)

         t = (ngrdpt - 1) * deltat

         tol = 1.0d-10

c        propagation time (seconds)

         tof = 86400.0d0 * t

         call twobody2(smu, tol, tof, rti1, vti1, rtf1, vtf1)

c        load data array (au & au/day)

         z(1) = rtf1(1) / aunit
         z(2) = rtf1(2) / aunit
         z(3) = rtf1(3) / aunit

         z(4) = vmult * vtf1(1)
         z(5) = vmult * vtf1(2)
         z(6) = vmult * vtf1(3)
```

```
      end if

      if (iphase .eq. 2) then

c        ******************************************************************
c        create state vector on second leg of transfer orbit at grid points
c        ******************************************************************

c        ---------------------------------------------------
c        solve lambert's problem for second heliocentric leg
c        ---------------------------------------------------

c        posigrade transfer

         direct = 1.0d0

c        less than one rev transfer

         revmax = 0.0d0

c        process first (and only) solution

         isn = 1

         tof1 = 86400.0d0 * (xjdate3 - xjdate2)

         call lambfunc(smu, rp2, rp3, tof1, direct, revmax,
     &                 statev, nsol)

         if (nsol .eq. 0) then

            pause 'nsol = 1'

            iferr = -1

            return

         end if

c        initial position vector of second heliocentric leg (km)

         do i = 1, 3

            rti2(i) = rp2(i)

         end do

c        extract initial velocity vector of second heliocentric leg (km/sec)

         vti2(1) = statev(4, isn)
         vti2(2) = statev(5, isn)
         vti2(3) = statev(6, isn)

         deltat = (xjdate3 - xjdate2) / (ngrid2 - 1)

         t = td1 + (ngrdpt - 1) * deltat

         tol = 1.0d-10

c        propagation time (seconds)

         tof = 86400.0d0 * (t - td1)

         call twobody2(smu, tol, tof, rti2, vti2, rtf2, vtf2)

c        load data array (au & au/day)
```

```
         z(1) = rtf2(1) / aunit
         z(2) = rtf2(2) / aunit
         z(3) = rtf2(3) / aunit

         z(4) = vmult * vtf2(1)
         z(5) = vmult * vtf2(2)
         z(6) = vmult * vtf2(3)

      end if

      return
      end
```

The following Fortran subroutine illustrates the computations necessary for evaluating the *right hand side* differential equations of motion.

```
      subroutine oderhs(iphase, t, ydyn, ny, p, np, frhs, nf, iferr)

c     computes the right hand sides of the
c     differential-algebraic (dae) equations

c     n-body heliocentric equations of motion

c     input

c      t       = simulation time (days)
c      ydyn(1) = x-component of heliocentric position vector
c      ydyn(2) = y-component of heliocentric position vector
c      ydyn(3) = z-component of heliocentric position vector

c      ydyn(4) = x-component of heliocentric velocity vector
c      ydyn(5) = y-component of heliocentric velocity vector
c      ydyn(6) = z-component of heliocentric velocity vector

c     output

c      frhs = right hand sides of differential-algebraic system

c     note: unit of position vector is astronomical units and
c           unit of components of velocity vector is au/day

c     ***********************************

      implicit double precision (a-h, o-z)

      include 'socscom1.inc'

      integer iphase, ny, np, nf, iferr

      dimension ydyn(ny), p(np), frhs(nf), vplanet(3), vtmp(3)

      dimension rplanet(3), rp(9, 3), rp2sc(9, 3), rp2scm(9)

      dimension q(9), f(9), d3(9), asun(3), accp(3), asrp(3)

c     initialize evaluation error flag

      iferr = 0

c     -------------------------------------
c     central body (sun) point mass gravity
c     -------------------------------------

c     distance from sun to spacecraft
```

```
      rs2scm = sqrt(ydyn(1) * ydyn(1) + ydyn(2) * ydyn(2)
     &         + ydyn(3) * ydyn(3))

      do i = 1, 3

         asun(i) = -wmu(1) * ydyn(i) / rs2scm**3

      end do

c     ----------------------------
c     planetary point mass gravity
c     ----------------------------

      if (iephtype .eq. 1) then

         nplanets = 8

      else

         nplanets = 9

      end if

      xjdate = xjdatei1 + t

c     calculate heliocentric position vectors of each planet

      do i = 1, nplanets

         call p2000(i, xjdate, rplanet, vplanet)

         do j = 1, 3

            rp(i, j) = rplanet(j) / aunit

         end do

      end do

c     compute planetocentric position vectors of spacecraft

      do i = 1, nplanets

         rp2sc(i, 1) = ydyn(1) - rp(i, 1)

         rp2sc(i, 2) = ydyn(2) - rp(i, 2)

         rp2sc(i, 3) = ydyn(3) - rp(i, 3)

      end do

c     compute planetocentric position magnitudes of spacecraft

      do i = 1, nplanets

         rp2scm(i) = sqrt(rp2sc(i, 1)**2 + rp2sc(i, 2)**2
     &                    + rp2sc(i, 3)**2)

      end do

c     compute f(q) and other functions

      do k = 1, nplanets

         do i = 1, 3

            vtmp(i) = ydyn(i) - 2.0d0 * rp(k, i)
```

```
            end do

            call vdot(ydyn, vtmp, dot1)

            dot2 = rp(k, 1) * rp(k, 1) + rp(k, 2) * rp(k, 2)
     &           + rp(k, 3) * rp(k, 3)

            q(k) = dot1 / dot2

            f(k) = q(k) * ((3.0d0 + 3.0d0 * q(k) + q(k) * q(k))
     &           / (1.0d0 + (1.0d0 + q(k))**1.5d0))

            d3(k) = rp2scm(k) * rp2scm(k) * rp2scm(k)

         end do

         do j = 1, 3

            accp(j) = 0.0d0

            do k = 1, nplanets

               if ((k .ne. ip1) .and. (k .ne. ip2)
     &                          .and. (k .ne. ip3)) then

                  accp(j) = accp(j) - wmu(k + 1)
     &                    * (ydyn(j) + f(k) * rp(k, j)) / d3(k)

               end if

            end do

         end do

c        evaluate right hand sides of differential equations

         frhs(1) = ydyn(4)

         frhs(2) = ydyn(5)

         frhs(3) = ydyn(6)

         frhs(4) = asun(1) + accp(1)

         frhs(5) = asun(2) + accp(2)

         frhs(6) = asun(3) + accp(3)

         return
         end
```

This subroutine illustrates typical *SOS* compatible point function calculations.

```
         subroutine odepf(iphase, iphend, time, ydyn, nydyn, parm,
        &                 nparm, ptf, nptf, iferr)

c        evaluate phase dependent "position & velocity matching"
c        point functions and scalar delta-v objective function

c        **********************************

         implicit double precision (a-h, o-z)

         include 'socscom1.inc'
```

```
      parameter (zero = 0.0d0, one = 1.0d0)

      dimension ydyn(nydyn), parm(nparm), ptf(nptf)

      dimension rp(3), vp(3), rp2sc(3), vp2sc(3), rhat(3), vhat(3)

      dimension vinf_in(3), vinf_out(3), v1xv2(3)

      data pi / 3.141592653589793d0 /

      data rtd / 57.295779513082323d0 /

c     initialize evaluation error flag

      iferr = 0

      if (iphase .eq. 1 .and. iphend .eq. -1) then

c        ********************************************************************
c        launch planet "position & velocity match" at the beginning of phase 1
c        ********************************************************************

c        current julian date

         xjdate = xjdatei1 + time

c        compute launch planet position (km) and velocity (km/sec) vectors

         call p2000(ip1, xjdate, rp, vp)

c        position vector match point functions (au)

         ptf(1) = ydyn(1) - rp(1) / aunit

         ptf(2) = ydyn(2) - rp(2) / aunit

         ptf(3) = ydyn(3) - rp(3) / aunit

c        velocity vector match point functions (au/day)

         ptf(4) = ydyn(4) - (vmult * vp(1) + parm(1))

         ptf(5) = ydyn(5) - (vmult * vp(2) + parm(2))

         ptf(6) = ydyn(6) - (vmult * vp(3) + parm(3))

         if (iopt .eq. 1 .or. iopt .eq. 3) then

c           ************************************************
c           launch delta-v contribution to objective function
c           ************************************************

            ptf(7) = sqrt(parm(1)**2 + parm(2)**2 + parm(3)**2)

         end if

      end if

      if (iphase .eq. 1 .and. iphend .eq. +1) then

c        ***********************************************
c        flyby planet position vector match and incoming
c        v-infinity point functions at the end of phase 1
c        ***********************************************

c        current julian date
```

```
            xjdate = xjdatei1 + time
c           compute launch planet position (km) and velocity (km/sec) vectors
            call p2000(ip2, xjdate, rp, vp)
c           position vector match point functions (au)
            ptf(1) = ydyn(1) - rp(1) / aunit
            ptf(2) = ydyn(2) - rp(2) / aunit
            ptf(3) = ydyn(3) - rp(3) / aunit
c           compute incoming v-infinity vector (km/sec)
            do i = 1, 3
               vinf_in(i) = ydyn(i + 3) / vmult - vp(i)
            end do
c           load incoming v-infinity vector point functions
            ptf(4) = vinf_in(1)
            ptf(5) = vinf_in(2)
            ptf(6) = vinf_in(3)
c           scale incoming v-infinity vector point functions
            ptf(4) = 1.0d-6 * ptf(4)
            ptf(5) = 1.0d-6 * ptf(5)
            ptf(6) = 1.0d-6 * ptf(6)
         end if
         if (iphase .eq. 2 .and. iphend .eq. -1) then
c           ****************************************************************
c           "v-infinity and flyby altitude match" at the beginning of phase 2
c           ****************************************************************
c           current julian date
            xjdate = xjdatei1 + time
c           compute flyby planet position (km) and velocity (km/sec) vectors
            call p2000(ip2, xjdate, rp, vp)
c           load incoming v-infinity vector from parameter vector (km/sec)
            do i = 1, 3
               vinf_in(i) = parm(i + 3)
c              scale point functions
               ptf(i) = 1.0d-6 * parm(i + 3)
            end do
c           compute outgoing v-infinity vector (km/sec)
```

```
            do i = 1, 3

               vinf_out(i) = ydyn(i + 3) / vmult - vp(i)

            end do

c           calculate v-infinity magnitudes (km/sec)

            vinfm_in = vecmag(vinf_in)

            vinfm_out = vecmag(vinf_out)

c           ---------------------------
c           v-infinity match constraint
c           ---------------------------

            ptf(4) = vinfm_in - vinfm_out

c           scale point function

            ptf(4) = 1.0d-2 * ptf(4)

            call vcross(vinf_in, vinf_out, v1xv2)

            v1xv2m = vecmag(v1xv2)

c           true anomaly at infinity (radians)

            phi1 = 0.5d0 * pi + 0.5d0 * asin(v1xv2m
     &           / (vinfm_in * vinfm_out))

c           calculate planet-centered hyperbolic orbital elements

            fbecc = -1.0d0 / cos(phi1)

            fbsma = -pmu(ip2) / (vinfm_in * vinfm_in)

            fbrp = fbsma * (1.0d0 - fbecc)

c           actual flyby altitude (kilometers)

            fbaltitude = fbrp - rep(ip2)

c           -------------------------------
c           flyby altitude match constraint
c           -------------------------------

            ptf(5) = fbaltitude - fbalt

c           scale constraint

            ptf(5) = 1.0d-4 * ptf(5)

         end if

         if (iphase .eq. 2 .and. iphend .eq. +1) then

c           ****************************************************************
c           arrival planet "position & velocity match" at the end of phase 2
c           ****************************************************************

c           current julian date

            xjdate = xjdatei1 + time

c           compute arrival planet position (km) and velocity (km/sec) vectors
```

```
         call p2000(ip3, xjdate, rp, vp)

c        position vector match point functions (au)

         ptf(1) = ydyn(1) - rp(1) / aunit

         ptf(2) = ydyn(2) - rp(2) / aunit

         ptf(3) = ydyn(3) - rp(3) / aunit

         if (itraj .eq. 2) then

c           velocity vector match point functions (au/day)

            ptf(4) = ydyn(4) - (vmult * vp(1) + parm(1))

            ptf(5) = ydyn(5) - (vmult * vp(2) + parm(2))

            ptf(6) = ydyn(6) - (vmult * vp(3) + parm(3))

         end if

         if (iopt .eq. 2 .or. iopt .eq. 3) then

c           ************************************************
c           arrival delta-v contribution to objective function
c           ************************************************

            ptf(7) = sqrt(parm(1)**2 + parm(2)**2 + parm(3)**2)

         end if

      end if

      return
      end
```

The following is the source code for a typical *SOS* support subroutine. This Fortran code implements Shepperd's method for solving the two-body initial value problem (IVP).

```
      subroutine twobody2 (xmu, tol, tau, ri, vi, rf, vf)

c     solution of the two body initial value problem

c     Shepperd's method

c     input

c      xmu = gravitational constant (km**3/sec**2)
c      tol = convergence tolerance
c      tau = propagation time interval (seconds)
c      ri  = initial eci position vector (kilometers)
c      vi  = initial eci velocity vector (km/sec)

c     output

c      rf = final eci position vector (kilometers)
c      vf = final eci velocity vector (km/sec)

c     ***********************************

      implicit double precision (a-h, o-z)

      dimension ri(3), vi(3), rf(3), vf(3)
```

```
      data pi2 /6.283185307179586d0/

      call vdot(ri, vi, rdotv)

      rm1 = vecmag(ri)

      call vdot(vi, vi, visq)

      beta = (2.0d0 * xmu / rm1) - visq

      u = 0.0d0

      if (beta .ne. 0.0d0) then

         umax = 1.0d0 / sqrt(abs(beta))

      else

         umax = 1.0d+24

      end if

      umin = -umax

      if (beta .le. 0.0d0) then

         delu = 0.0d0

      else

         p = pi2 * xmu * beta**(-1.5)

         n = dint((1.0d0 / p) * (tau + 0.5d0 * p
     +       - 2.0d0 * rdotv / beta))

         delu = pi2 * n * beta**(-2.5)

      end if

      tsav = 1.0d99
      niter = 0

c     kepler iteration loop

      do while (.true.)

         niter = niter + 1

         q = beta * u * u

         q = q / (1.0d0 + q)

         u0 = 1.0d0 - 2.0d0 * q

         u1 = 2.0d0 * u * (1.0d0 - q)

c        continued fraction iteration

         n = 0
         l = 3
         id = 15
         k = -9
         a = 1.0d0
         b = 1.0d0
         g = 1.0d0

         do while (.true.)
```

```
            gsav = g
            k = -k
            l = l + 2
            id = id + 4 * l
            n = n + (1 + k) * l

            a = id / (id - n * a * q)
            b = (a - 1.0d0) * b
            g = g + b

            if (abs(g - gsav) .lt. tol) exit

         end do

         uu = (16.0d0/15.0d0) * u1 * u1 * u1 * u1 * u1 * g + delu

         u2 = 2.0d0 * u1 * u1
         u1 = 2.0d0 * u0 * u1
         u0 = 2.0d0 * u0 * u0 - 1.0d0
         u3 = beta * uu + u1 * u2 / 3.0d0

         r = rm1 * u0 + rdotv * u1 + xmu * u2

         t = rm1 * u1 + rdotv * u2 + xmu * u3

         dtdu = 4.0d0 * r * (1.0d0 - q)

c        check for time convergence

         if (abs(t - tsav) .lt. tol) exit

         usav = u
         tsav = t
         terr = tau - t

         if (abs(terr) .lt. abs(tau) * tol) exit

         du = terr / dtdu

         if (du .lt. 0.0d0) then

            umax = u

            u = u + du

            if (u .lt. umin) u = 0.5d0 * (umin + umax)

         else

            umin = u

            u = u + du

            if (u .gt. umax) u = 0.5d0 * (umin + umax)

         end if

c        check for independent variable convergence

         if (abs(u - usav) .lt. tol) exit

c        check for more than 50 iterations

         if (niter .gt. 50) then

            pause ' more than 50 iterations in twobody2'
```

```
            return

         end if

      end do

      uc = u

      fm = -xmu * u2 / rm1
      ggm = -xmu * u2 / r
      f = 1.0d0 + fm
      g = rm1 * u1 + rdotv * u2
      ff = -xmu * u1 / (rm1 * r)
      gg = 1.0d0 + ggm

c     compute final state vector

      do i = 1, 3

         rf(i) = f * ri(i) + g * vi(i)

         vf(i) = ff * ri(i) + gg * vi(i)

      end do

      return
      end
```